A HEURISTIC METHOD FOR REDUCING MESSAGE REDUNDANCY IN A FILE TRANSFER ENVIRONMENT

by

William Robert Bodwell

Dissertation submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
Electrical Engineering

APPROVED:

A. W. Bennett, Chairman

F. G. Gray

F. L. Lam

R. A. Thompson

W. F. VanLandingham

March, 1976

Blacksburg, Virginia

ACKNOWLEDGMENT

The author would like to extend his appreciation to his graduate committee for their many comments and suggestions during the course of progress of this dissertation. A special thanks goes to my program chairman, who patiently guided me to the selection of the chosen dissertation topic.

The investigation would have been impossible without the support of Newport News Shipbuilding. I would like to thank for providing me with challenging work assignments which justified the use of computer facilities without which the investigation could not have been completed. The author also thanks for initially entering some of the dissertation draft into the computer.

The greatest thanks of all goes to my wife, and two children, for their patience and understanding during the years of study required for earning the doctoral degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Computer communications have grown rapidly in recent years because of user demand and important technological developments, e.g., microwave technology and satellite technology. Favorable Federal Communication Commission (FCC) decisions have allowed innovative applications of this new technology. Viable computer networks are being developed using facilities supplied by the common-carriers (AT&T, Western Union, etc.), and the developing specialized common-carriers (Datran, Telenet, etc.). The computer networks allow sharing of resources such as CPU programs, databases, memory space, personnel, and long-haul links.

A computer network can provide an economical and efficient means by which information can be exchanged between hosts. As the amount of information to be exchanged has increased, the most prevalent means for providing the increased capability has been to increase the channel's transmission speed [65]. This research concerns itself with the development of techniques for minimizing the amount of information which must be transferred between hosts to reconstruct file content. Encoded information can be transmitted over a communication channel in a shorter time

period, or transmitted via a channel with a smaller bandwidth (bandwidth compression), or stored in a smaller disk area. Application of techniques developed in this dissertation will provide the potential of more economical means of transmitting information between hosts, and of storing the information on disk in a compressed, secure form.

REASONS FOR NETWORK DEVELOPMENT

A computer network can be defined as an interconnected group of independent host computers that automatically communicate with one another and can share resources. Each host has the capability of being used in a local mode and also participating in network activities [26,29]. The communication subnetwork interconnects the host machines using either leased lines and/or other facilities available from either the common-carriers and/or specialized common-carriers. General software considerations for communication processors are described by Chyzik and Mills [21,68]. The communication processors on the Advanced Resource Agency Network (ARPANET) are the Interface Message Processor (IMP) described by Heart, and the Terminal Interface Processor (TIP) described by Ornstein [45,70].

A computer network can provide an economical and efficient means by which information (programs, data, etc.)

and special equipment (plotters, graphic terminals, etc.) of one host might be shared by network users. A database can be either centralized or distributed among nodes as system characteristics warrant; network users still have access to the totality of information [11,26]. Farber discussed the file structure of a distributed computer system [36]. Chu and Whitney addressed the problem of optimal location of the individual files which comprise the database [20,93]. Booth outlines database usage considerations [14]. Usage of a minicomputer to access the file system of a batch-oriented computer was the dissertation topic of Ives [51]. A simple protocol to handle file transfers was developed for the TENEX operating system [13]. Thomas reported on a resource sharing executive, RSEXEC, which facilitates network file usage and runs on TENEX host computers on the ARPANET [86]. Cerf described an experimental service for adaptable data reconfiguration which overcomes simple data format incompatibilities of heterogeneous hosts on the ARPANET [18]. The most advanced stage of EDP applications, database applications, has the potential of returning economic benefits, and communication between information sources/users is of much greater importance [94].

A computer network frequently reduces communication line costs through improved efficiency in the use of data communication facilities. Cost reductions result from

economy of scale, i.e., generally lower unit costs through use of higher bandwidth lines, as well as through improved utilization of the leased circuits [63].

At present, load-balancing is the most advanced stage of network usage and it offers the best, lowest-cost utilization of computer and communication resources. Load-balancing implies rapid, reliable, and cost-effective transfer of programs and data between nodes of the network.

Roberts indicated two additional advantages of networking: increased communication between computer researchers, and measurement of efficient ways to utilize computers [79].

EVOLUTION OF COMPUTER NETWORKS

Computer networks are part of a natural evolution in telecommunication-oriented systems. Telecommunication networks were originally developed for timesharing and remote job entry (RJE).

Early History

One of the earliest attempts at interconnecting a large number of computers was the Semi-Automatic Ground Environment (SAGE) air defense system which became operational in 1958. SAGE was used by the military to analyze and display radar data from sensors scattered over the continent [60].

The American Airlines' Reservation System, SABRE, was developed in the 1950's, and became available on a commercial basis in the early 1960's as what was then the world's largest real-time commercial application [56]. Societe' Internationale de Telecommunications Aeronatiques (SITA) is a packet-switched network, in operation since 1966, which handles 175 international airlines' reservations (enquiry/response) and administrative matters such as lost baggage, flight servicing information, etc. In 1973, SITA handled 255 million messages which is believed to exceed the volume handled by ITT, Western Union International, and RCA Globcom combined [47].

Automatic Digital Network (AUTODIN) was developed in 1963 to improve military data communications. The design of AUTODIN was heavily influenced by reliabilility considerations [60].

Recent History

Probably the most ambitious network design ever undertaken is the Advanced Research Agency Network (ARPANET). ARPANET has had such an important impact upon data communications that it will be described separately in the next section.

CYBERNET, DCS, MERIT, OCTOPUS, TSS, and TUCC are computer networks which have been described by Doll and Farber [26,33]. CYBERNET is a distributed, heterogeneous

commercial network which utilizes a broad spectrum of switched, leased, and satellite communication facilities. DCS is an experimental computer network being developed and constructed at the University of California at Irvine which is based on a digital communication ring topology utilizing the Bell System T1 technology and fixed-length messages. MERIT is a joint cooperative effort between Michigan State University, Wayne State University, and the University of Michigan which was established to facilitate the sharing of the computer resources of the member schools. OCTOPUS is a heterogeneous network developed at the Lawrence Berkeley Laboratory of the University of California. This network consists of CDC worker machines operating as time-shared facilities interconnected via 12-megabit capacity hardwired cables. Users interface to the network through PDP-8's, and are able to access all files through the workers, i.e., CDC machines that are connected to dual PDP-10's which maintain the file system. TSS is a homogeneous, distributed network cooperative venture between IBM and some of its 360/67 customers. The Triangle University Computational Center (TUCC) Network is a joint undertaking of the Duke, North Carolina State, and North Carolina Universities. TUCC is a centralized, homogeneous network interconnecting four IBM 360 facilities.

ARPANET

The Advanced Research Project Agency Network (ARPANET) was established in late 1969 to provide a resource-sharing capability between a number of geographically distributed, heterogeneous computer installations. ARPANET has been extensively reported in the literature [1, 5, 13, 15, 17, 18, 22, 23, 24, 39, 40, 42, 45, 53, 54, 70, 76, 78, 82]. The ARPANET became available for useful resource-sharing in 1971. Today the ARPANET stretches from Hawaii to Europe and encompasses approximately 50 nodes as shown in Figures 1 and 2. Users may access the network either via a host computer or ARPA Network Terminal System (ANTS) which is connected to either an Interface Message Processor (IMP) or Terminal Interface Processor (TIP) as shown in Figure 3 [15,45,70]. Terminal users may interface the network directly via a TIP [10,70]. The IMP's and TIP's are minicomputers interconnected via 50,000 bits/second lines. The network uses store-and-forward (S/F) packet-switching with distributed control for transferring messages between hosts [42]. Messages are accepted by the IMP's, divided into packets, dynamically routed over the communication subnetwork, packets reassembled into messages, and messages delivered to the receiving host [40].

The network fulfilled the following major design

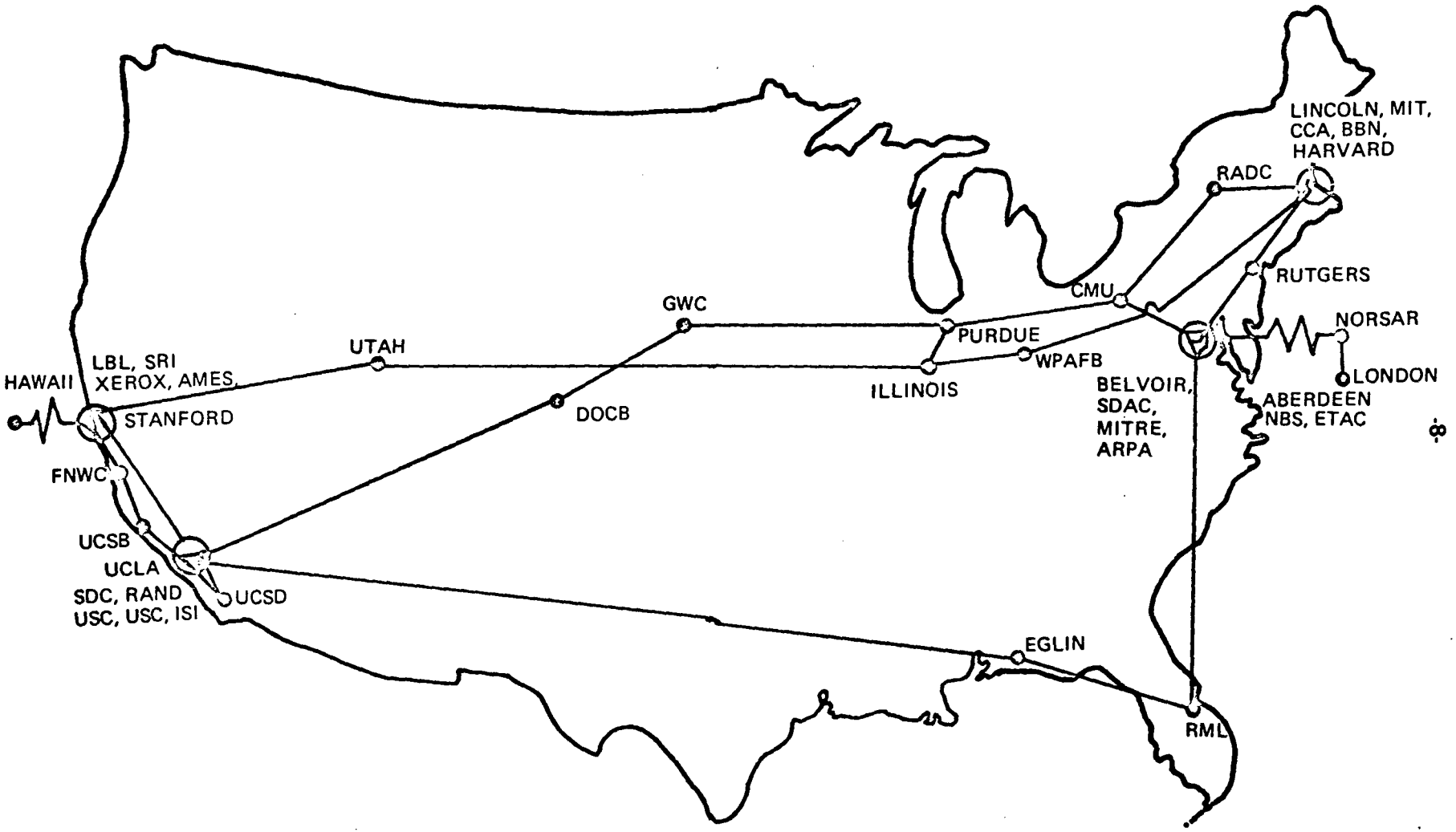

FIGURE 1. GEOGRAPHIC MAP OF THE ARPANET, JANUARY 1975

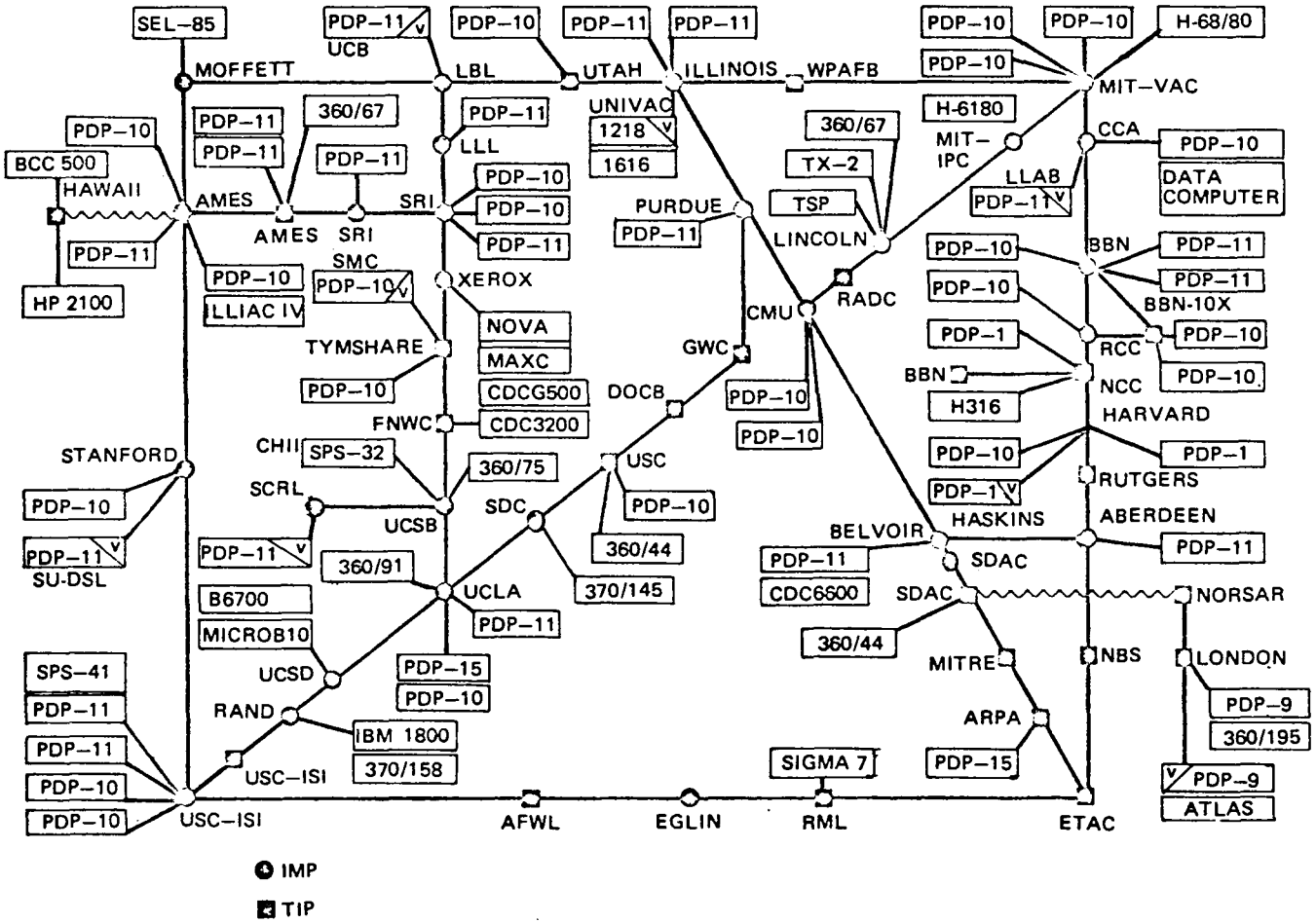

FIGURE 2. LOGICAL MAP OF THE ARPANET, JANUARY 1975

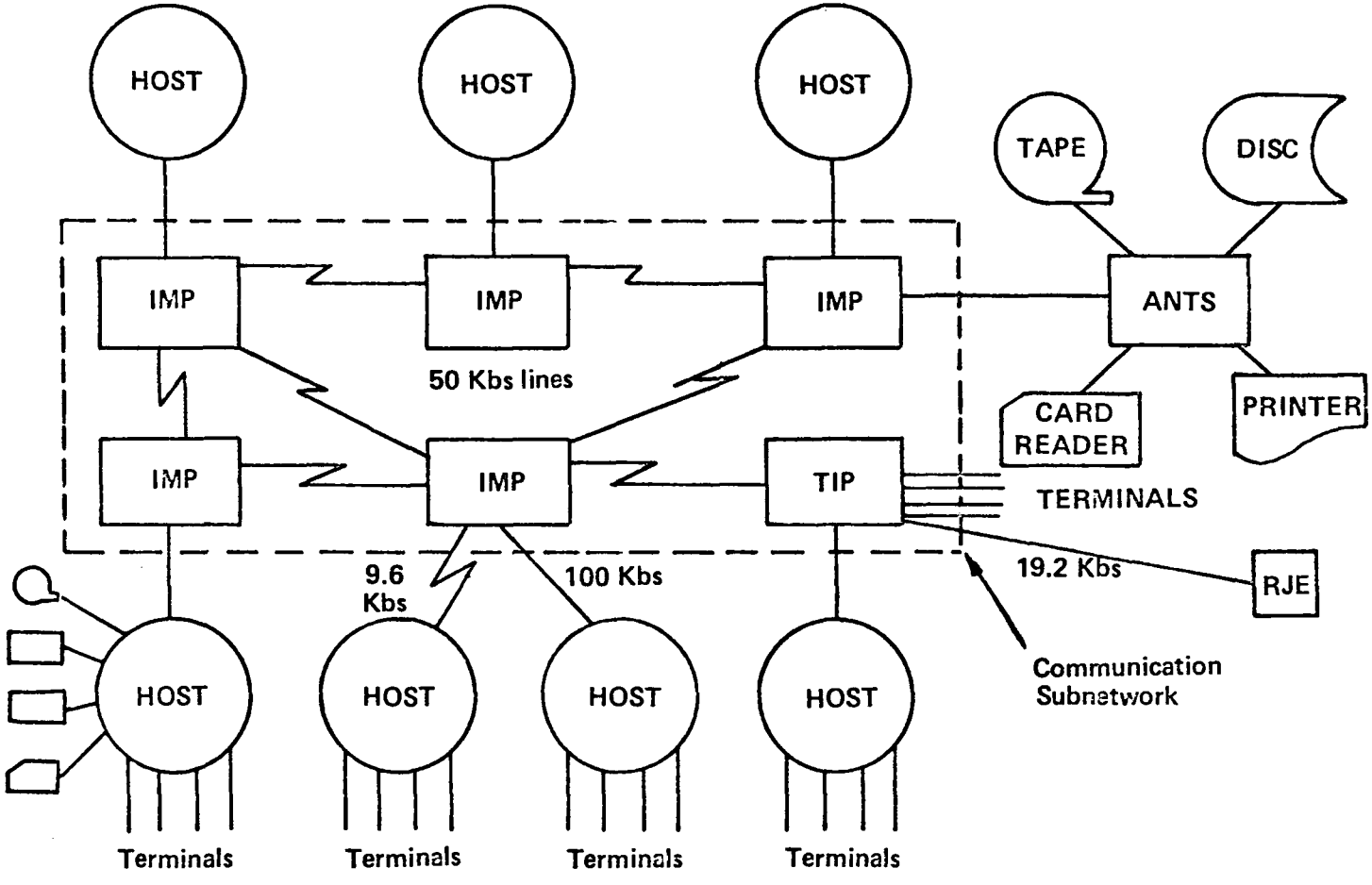

FIGURE 3. ARPANET ACCESS

goals:

1. Permitted network users to share resources at remote nodes in an interactive mode.
2. Developed a highly reliable and economically effective digital communication subnetwork.
3. Permitted broad access to unique and powerful facilities, e.g., ILLIAC IV.

Cole and Frank reported on the measurement and evaluation of the performance of the ARPANET [22,39]. A user must explicitly specify which host should process each subtask or activity of a complete network job in a meaningful and sensible manner, e.g., PDP-10 for interactive timesharing, ILLAC IV for large-scale processing, etc. The network has obtained a highly reliable subnetwork with the expectation that less than one undetected error will occur per year per node. Crother discusses overall network reliability [24]. In addition, the network has been used for the bulk movement of large data and program files in preference to mailing tapes since the network is more reliable, much faster, and usually less costly because of the elimination of human handling [77].

An analysis of March 1973 network usage reveals 2 million dollars per year was being spent on computing resources accessed through the network, but the expenditure of 6 million dollars would have been necessary for these

resources if the network did not exist. The 4 million dollars saving more than offsets the 3.5 million dollars annual cost of the communication lines, modems, and amortization of the network's IMP's and TIP's [76,82].

As mentioned earlier, the ARPANET is heterogeneous. The 50 host computers consist of 35 different types. Three minicomputers have been used as IMP's: Honeywell 516 (1970; .168), Honeywell 316 (1971; .102), and Lockheed Sue (1974; .026). The numbers within parentheses refer to the year of introduction and the cost in dollars of processing a thousand packets. In general, the cost per unit computing capability for both host's and IMP's has been decreasing by a factor of 10 every five years. Unfortunately, data transmission services available from the Bell System have been decreasing by a factor of 10 every 22 years. For the ARPANET, the cost of processing 1000 packets equaled the expense of transporting 1000 packets in 1969; thus, in 1974 communication costs were five (5) times the processing costs per thousand packets. The introduction of satellites as communication nodes offers the potential for significant reduction in communication costs since satellite technology is decreasing costs for a level of service by a factor of 10 every 6.7 years. Figure 4 indicates the trends of computing, terrestrial communication costs, and satellite communication costs based on packet-switching technology [76]. The trends

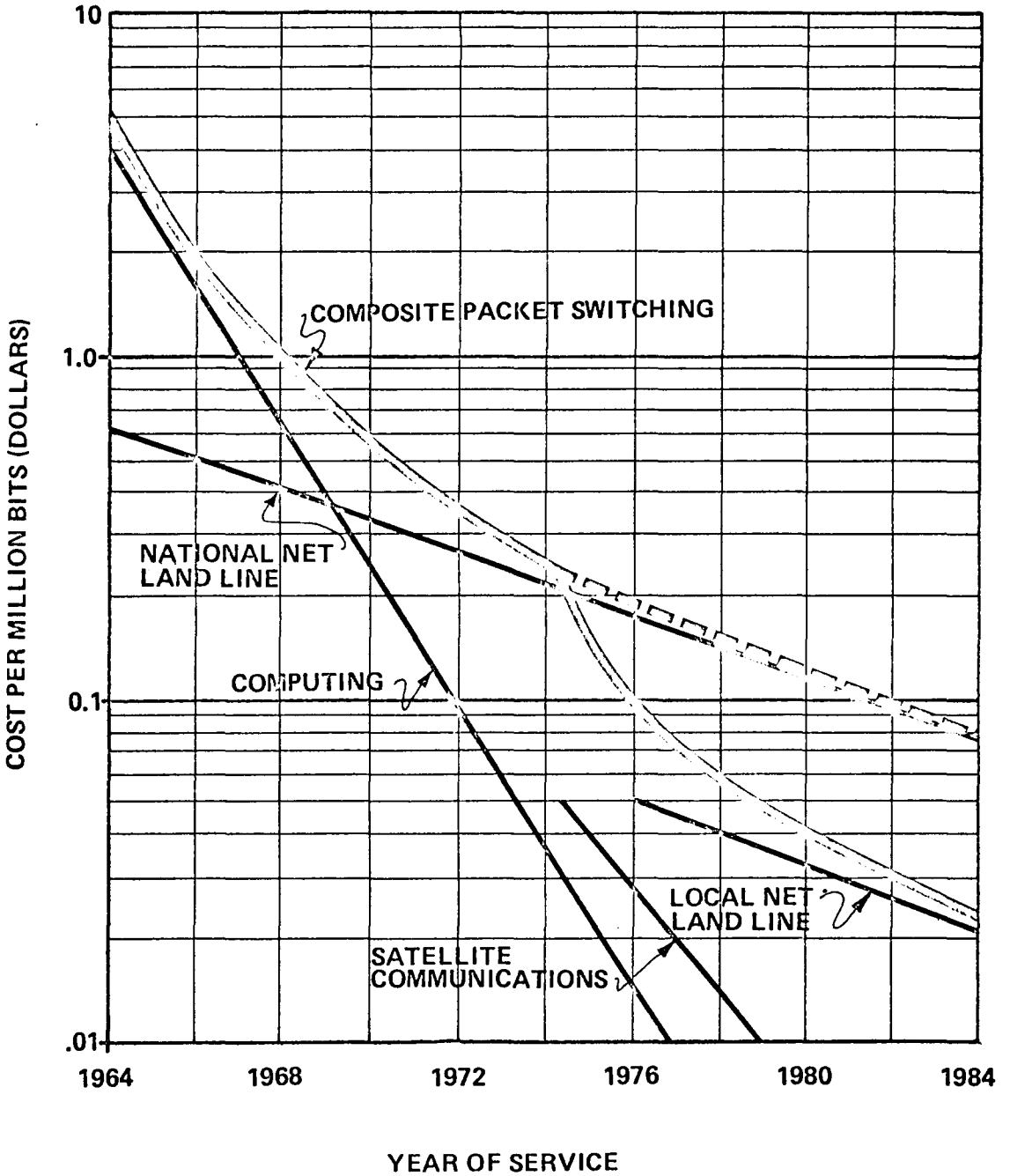

FIGURE 4. COST/MILLION BITS VS YEAR OF SERVICE USING PACKET–SWITCHING TECHNIQUES

shown in Figure 4 should be applicable to Telenet's commercial value-added network.

EVOLUTION OF COMMON-CARRIERS

The services which a common-carrier can offer must be approved by the Federal Communication Commission (FCC). Advances in technology coupled with favorable decisions by the FCC have led to the offering of new services by the common-carriers and the formation of specialized common-carriers. To understand the reasons for the sudden emergence of, and the increased importance of, data communications, one must review the major advances in technology and FCC decisions.

Development of microwave radio technology and satellite technology have provided a significant advance in the state-of-the-art of information transmission. The rapid expansion of computer applications requiring data communications has resulted in a growing requirement of services not traditionally provided by the telephone and telegraph companies; thus, technological advances and a ready market for services that the technology could provide have led to the creation of specialized common-carriers. The specialized common-carriers petitioned the FCC for permission to operate since the telecommunications industry is regulated.

Since the mid-1960's, the FCC has been faced with four problem areas:

1. Unacceptable data communications by the traditional common-carriers.
2. Interconnection of foreign equipment to common-carriers' equipment.
3. Technological advances, e.g., microwave technology and satellite technology.
4. Pricing structure for data transmission.

In November 1966, the FCC issued a Notice of Inquiry which sought information from interested parties on computer-communication policy. In June 1968, the FCC issued the Carterfone decision which authorized the interconnection of customer equipment to common-carrier equipment. The FCC in August 1969, approved the first specialized common-carrier by granting Microwave Communications, Inc. (MCI) a construction permit after nearly six years of litigation. The FCC issued the First Report and Order during March 1971 which gave approval to the specialized carrier concept after more than four years of proceedings. In June 1972, the FCC issued the ruling which permitted all qualified applicants to provide domestic communication satellite service [62]. In November 1973, the FCC authorized Packet Communications, Inc. (PCI) to construct and operate a commercial value-added-communication network (VAN) based on

packet-switching [29]. Telenet was authorized to construct and operate a VAN in April 1974 [84]. McKee and McCormick have proposed the creation of a hybrid teleprocessing industry to fill the gap caused by FCC's maximum separation policy, i.e., prohibition against more that an incidental amount of both data processing and communications [93].

Thus, technology and favorable rulings by the FCC have led to the creation of specialized common-carriers which will provide knowledgeable users with the means for reducing their computer-communication costs, improving existing services, implementing new services, etc. Minimization of costs for a certain level of service will be further complicated by new tariffs proposed by the traditional common-carriers.

STATEMENT OF PROBLEM BEING CONSIDERED

Through previous dissertation sections, an attempt has been made to show that during the 1970's intercomputer communications should increase significantly. The communication will often involve the transfer of files between hosts. This dissertation is directed towards the development of a technique which improves the efficiency with which this type of transfer can be accomplished. The specific goals of this dissertation were to:

1. Develop an encoding/decoding technique which

effectively reduces both informational and structural redundancy. The Substitution Method is based on the premise that effective Encoding/Decoding Tables can be generated for a specified language by using the syntax of the language and statistics gathered by examining typical files written in the language.

2. Compare the informational data compression obtainable using the Shannon-Fano Method and Substitution Method with theoretical minimums based on fixed-to-variable length encoding.
3. Demonstrate the Substitution Method using typical JCL and Fortran source files. Input symbol frequencies for JCL and Fortran source files were obtained. Appropriate Shannon-Fano and Substitution Method Encoding/Decoding Tables for JCL and Fortran source files were generated.
4. Describe how the benefits of the Substitution Method of encoding/decoding might be used in a semi-optimal manner to compress data files.
5. Economically evaluate the Substitution Method when communication costs are based on tariffs filed by AT&T and some of the new specialized

common-carriers (American Satellite, Datran, Telenet).

RESEARCH SIGNIFICANCE

Computer communications have grown rapidly in recent years because of important technological developments and innovative applications of this new technology. Much of the future growth will be due to the development of viable computer networks. Usage of these networks will significantly increase the transfer of file information between the respective hosts. The transfer of files between hosts requires the faithful reproduction of the files at the receiving host; but, it does not imply that the input file be transferred bit-by-bit, if a more efficient technique is available.

This research undertakes development of an efficient method of encoding/decoding source files. The technique requires that a set of user files for each category of files be examined to ascertain the frequency distribution of symbols, fixed strings, and repeated symbol strings to determine symbol and structural redundancy. Information gathered during the examination of these files, when combined with the user created Source Language Syntax Table, is used to generate Encoding/Decoding Tables which allows the reduction of both symbol and structural redundancy. The

Encoding/Decoding Tables allow frequently encountered strings to be represented by only one or two symbols through the utilization of table shift symbols. The table shift symbols allow less frequently encountered symbols of the original alphabet to be represented as entries in a Secondary Encoding/Decoding Table. A technique is described which enables a programmer to easily modify his Fortran program such that he can take advantage of the Substitution Method's ability to compress data files by removing both informational and structural redundancy.

Obviously, the encoding/decoding overhead must be considered in evaluating the Substitution Method's usefulness. Each user file to be transferred is preprocessed (encoded) at cost, C[prep], to reduce both symbol and structural redundancy which need not be transferred for faithful reproduction of the file. The file is transferred over a noiseless channel at cost, C[ptran]. The channel consists of presently available or proposed services of the common-carriers and specialized common-carriers. The received file is post-processed (decoded) to reconstruct the original source file at cost, C[post]. The costs associated with pre-processing, transferring, and post-processing are compared with the cost, C[otran], of transferring the entire file in its original form.

The entire development of the Substitution Method has

been based upon the software structure of the Honeywell 6000 series of computers. The Substitution Method for reducing file symbol and structural redundancy should provide guidelines for similar operations on different series of computers.

Chapter 2

INFORMATION THEORY, DATA COMPRESSION TECHNIQUES, AND LANGUAGE GRAMMARS

Information theory deals with three basic concepts: the measure of information, the capacity of communication channels, and coding as a means of utilizing channels at full capacity [16]. Entropy is used as the measure of information. Ash provides an excellent discussion on methods of calculating channel capacity [7]. This dissertation will not concern itself with channel capacity since there exists certain standard rates for data transmission, e.g., 110 bits/second, 9,600 bits/second, etc. Coding involves the use of an encoder between the source and the channel which assigns code words that are (hopefully) more suitable for transmission through the channel. A decoder exists at the receiving end of the channel to return the original source. Data compression is obtained through "proper" encoding. Grammars are defined which are utilized to describe the syntax of the languages to be examined, e.g., JCL and Fortran source files.

INFORMATION THEORY

The mathematical basis for information theory was

presented by Shannon in The Mathematical Theory of Communication [81]. The fundamental theorem of information theory is as follows:

> "Given an information source and communication channel, there exists a coding technique such that the information can be transmitted over the channel at any rate less than the channel capacity with arbitrarily small frequency of errors despite the presence of noise." [16].

This dissertation is not concerned with noise since Value-Added-Networks (VAN's) will offer communication services which anticipate less than one undetected error per year per node. Abramson, Ash, and Reza have written texts which provide a much more detailed description of information theory [2,7,75]. The remainder of this dissertation reflects on the measure of information and coding which is utilized to match the source and the channel for maximum information transfer.

Entropy

Entropy, H[c0], as defined by Shannon, equals the average amount of information produced by a source, assuming statistical independence of the source. Conditional entropy, H [cn], equals the average information produced by a source when intersymbol dependence exists [81]. Nomenclature and equations are developed in Chapter 3 for tabulating symbol and string occurrences; calculating probability of symbol occurrences; calculating entropy, H[c0]; and calculating

conditional entropy, H[c1], based upon knowledge of one preceding symbol.

Entropy is calculated by summing the self-information contained within each symbol that a discrete source may produce when symbols are statistically independent. Let the source have an alphabet of size m. For each symbol i, where $1 \leq i \leq m$, there is a probability, p[i], that the symbol occurs. Entropy of a discrete source is calculated as:

$$H[c0] = \sum_{i=1}^{m} p[i] \log[2] (1/p[i]) \quad \text{bits/symbol} \quad (2\text{-}1)$$

where subscripts and logarithmic base are denoted by brackets, i.e., [ ]. Entropy is a statistical or ensemble average not designed around a particular message, but rather all possible messages. It is assumed that the source is stationary or ergodic so that the time and ensemble averages are identical.

Conditional Entropy

Discrete sources, such as files, are often constrained by certain rules which restrict the selection of successive symbols. The resulting intersymbol influence reduces the amount of uncertainty of the next symbol, and thus reduces the amount of information produced. Conditional entropy, H[cn], is used to account for this effect.

Conditional entropy is calculated by considering the entire history of preceding symbols [16,81]. Conditional entropy based upon knowledge of one preceding symbol can be calculated by:

$$H[c1] = \sum_{i=1}^{m} \sum_{j=1}^{m} p[i]\ p(j|i)\ \log[2]\ (1/p(j|i))$$

bits/symbol (2-2)

where $p(j|i)$ represents the conditional probability of symbol j occuring, given that symbol i has occurred. But, the joint probability, $p(i,j)$, can be expressed as:

$$p(i,j) = p[i]\ p(j|i) \qquad (2\text{-}3)$$

and,

$$\log[2]\ A = \log[10]\ A\ /\ \log[10]\ 2 \qquad (2\text{-}4)$$

thus,

$$H[c0] = 3.322 \sum_{i=1}^{m} p[i]\ \log[10]\ (1/p[i])$$

bits/symbol (2-5)

and,

$$H[c1] = 3.322 \sum_{i=1}^{m} \sum_{j=1}^{m} p(i,j)\ \log[10]\ (p[i]/p(i,j))$$

bits/symbol (2-6)

Conditional entropy of the English language was investigated by Shannon [80]. His study was based upon the standard telegraph language of 26 letters plus a space. A summary of his results are:

$$H[max] = \log[2]\ (27) = 4.75 \quad \text{bits/symbol}$$

$$H[c0] = 4.03 \quad \text{bits/symbol}$$

$$H[c1] = 3.32 \quad \text{bits/symbol}$$

$$H[c2] = 3.10 \quad \text{bits/symbol}$$

$$\cdot \cdot \cdot$$

$$H[c\infty] = .5\text{-}1.0 \quad \text{bits/symbol}$$

where, H[max] assumes equal probability of occurrence for each symbol. H[cn] is calculated by considering conditional probabilities based upon knowledge of n previous symbols. $H[c\infty]$ implies that with suitable coding, printed English could theoretically be transmitted in binary form with an average of one binary digit/symbol. Ascher and Nagy have demonstrated methods for compression of scan-digitized printed text [6].

Symbol Redundancy

A source with alphabet A which has m symbols contains redundant information if the probability of each symbol occurring is not 1/m (i.e., equally probable) and/or symbols which are not statistically independent. Symbol redundancy, R[sym], is calculated as:

$$R[sym] = 100\ (1 - H[cn]/H[max]) \quad \text{percent} \qquad (2\text{-}7)$$

where, H[cn] is entropy (n=o) or conditional entropy calculated with knowledge of n previous symbols, and H[max] assumes equal probability of occurrence for each symbol. H[cn]/H[max] is referred to as relative redundancy [16]. For a noiseless channel, redundancy in a message is undesirable and inefficient since the same information could be sent with fewer nonredundant symbols. Coding to reduce message redundancy is far more difficult than coding for error-detecting and error-correcting [69].

Structural and Total Redundancy

Files contain additional redundancy because of their structured nature, i.e., the presence of control/fill characters. Structural redundancy, R[str], is defined as:

$$R[str] = 100(n[cf]/n[t])\ (1-n[ncf]/n[cf])\ \text{percent} \quad (2\text{-}8)$$

Where,

n[cf] = Number of control/fill characters.

n[t] = Total number of characters.

n[ncf] = Number of necessary control/fill characters.

Thus, total file redundancy, R[total], for files can be expressed as:

$$R[total] = 100\ [(n[cf]/n[t])(1 - n[ncf]/n[cf]) + (1 - n[cf]/n[t])\ (1 - H[cn]/H[max])]\quad \text{percent} \quad (2.9)$$

DATA COMPRESSION TECHNIQUES

Data sources such as files contain a substantial amount of redundancy. Codes can be used to encode the source such that substantial data compression can be obtained, i.e., reduction of the average number of bits, nbar, required to represent a symbol. Encoding is a procedure for mapping a given set of messages[m[1],m[2],...,m[n]] onto a new set of encoded messages [c[1],c[2],...,c[n]] so that the transformation is one-to-one. The coding must be uniquely decipherable since the original message must be obtainable as the output of the decoder. As a further restriction, it is desirable that the code be instantaneous. Instantaneous, uniquely decipherable codes are discussed in detail in Ash [7].

There are three encoding methods which offer promise of data compression. The three possible encoding methods are:

1. Fixed-to-Variable Length Encoding
2. Variable-to-Fixed Length Encoding
3. Variable-to-Variable Length Encoding [8].

For fixed-to-variable length encoding, the noiseless encoding theorem says that: for a binary source,

$$\text{nbar} \geq H[c0] \qquad \text{bits/symbol} \qquad (2\text{-}10)$$

Where,

nbar equals the average symbol length

and,

H[c0] equals entropy.

For a binary source, using block encoding techniques, i.e., encoding more than one symbol a time,

$$nbar \geq H[cn] \quad \text{bits/symbol} \qquad (2\text{-}11)$$

Where,

H[cn] equals conditional entropy calculated with knowledge of n preceding symbols.

Thus, there are known limits on the size of nbar. Such equations do not exist for variable-to-fixed length encoding and variable-to-variable length encoding. A comparative evaluation of the three encoding methods as applicable to file compression for data transmission and storage is described.

Fixed-to-Variable Length Encoding -- Huffman Codes

Fixed-to-variable length encoding involves taking a source block composed of a fixed number of bits which is usually a symbol length and mapping the source block into a variable-length code word whose length is based upon the symbol's probability of occurrence. There are two widely acclaimed methods of fixed-to-variable length encoding: Huffman and Shannon-Fano [2,7,69,79]. Huffman coding is the

most efficient fixed-to-variable length encoding method [50]. The Shannon-Fano code is nearly as efficient as the Huffman code, but the Shannon-Fano code is much more easily developed. There are four potential areas of difficulty associated with fixed-to-variable length encoding. The first difficulty lies with the potential large difference in length of code words. There are various modified Huffman methods described in the literature which attempt to overcome this drawback [30,49,90]. The second problem area lies with fluctuations in the probabilities associated with the occurrence of each symbol. Conservative encoding techniques are described by Shaft, Smith, and Voorhis [79,83,90]. Data compression techniques using adaptive translation-permutation codes which accomplish sub-optimal data compression over varying file statistics are described by Felt [37]. Ott describes compact encoding of stationary Markov sources using an encoding scheme which utilizes the past output history of the source to adapt the encoding scheme to encode its future output compactly [71]. The third drawback lies with potential loss of synchronization over a communication channel should a character be received in error [30]. Ott suggests two means of overcoming synchronization problems: imbedding variable-length code words into an error correcting code and periodic retransmission of the transmitter's state vector and time

reference [71]. Unfortunately, both solutions require transmission of additional information. Another difficulty which must be considered in the potential utilization of significant amounts of computer resources in encoding and decoding code words.

Variable-to-Fixed Length Encoding

Variable-to-fixed length encoding involves taking a source block composed of a variable number of bits, usually a multiple of character size, and mapping the source block into a fixed-length code word. This is the most prevalent form of data compression in use today; it is used to compress repeated character strings. HASP, a protocol for data communications, uses a string-control byte (SCB) to compress data for transmission [85]. The SCB has the form:

1 K L J J J J J

Where,

K = 0 Implies Duplicate Character String.

K = 1 Implies No Duplicate Character String of LJJJJJ Following Characters.

L = 0 Implies JJJJJ Blank's.

L = 1 Implies Duplication of Next Character JJJJJ times.

3. Variable-to-Variable Length Encoding -- Golomb Codes

Variable-to-variable length encoding involves taking a source block composed of a variable number of bits or characters, and mapping the source block into a variable-length code word. Golomb used this method for run-length encoding [43]. Bahl and Kobayashi showed the superiority of Golomb codes over Huffman coding for compressing real images [7]. Variable-to-variable length encoding offers the potential of significant data compression and does not suffer from the defects which Huffman codes possess. For ease of encoding and decoding, code word lengths will be chosen to equal a multiple of 6-bits, i.e., BCD character size.

LANGUAGE GRAMMARS

Definitions and the metalanguage utilized to describe the syntax or structure of the language source files are defined in this section. The metalanguage to be used was first described by Chomsky and later modified by Backus [19]. The syntax of source language files are to be examined on a record-by-record basis where the record length is finite.

1. Definitions

Alphabet - Set of symbols whose length is finite. In this dissertation, two character were allowed: ASCII

(composed of 9 bits, and 128 combinations, i.e., two bits are always zero on the Honeywell 6080 for an ASCII character).

String - Finite sequence of symbols from the selected alphabet.

Record - Any string of finite length composed of strings and symbols from the alphabet. A record (sentence) may be the concatenation of one or more strings. Delimiters are themselves members of the alphabet.

Language - Any set of sentences over an alphabet. In this dissertation, two languages were evaluated: JCL and Fortran.

2. Nomenclature

$|n|$ - Length of string. String represented by small letters and symbols represented by capital letters.

$\langle\ \rangle$ - Angular brackets used to describe syntactic entries.

::= - May be composed of.

=> - Replaces, i.e., used when a syntactic entry is replaced by one of the strings of which it may be composed.

U ::= x | y is equivalent to U ::= x and U ::= y. This representation is called BNF which is an abbreviation of Backus-Normal Form.

U ::= a * b is equivalent to saying that U is the concatenation of strings a and b.

U ::= x */ y is equivalent to U is composed or x and/or y, i.e., take a choice of either or both.

m<U>n signifies that the variable string U must begin in position m of a record and end before or equal to position n.

m(U)n signifies that the fixed string U must begin in position m of a record and end before or equal to position n.

[U] Optional string U.

[U]n Optional string U may be repeated n times.

Chapter 3

DEVELOPMENT OF THE SUBSTITUTION METHOD OF ENCODING/DECODING FILES

The structure of source files is governed by certain rules which restrict the selection of successive symbols. The resulting intersymbol influence reduces the amount of uncertainty of the next symbol, and this reduces the amount of information produced. The Substitution Method reduces this redundancy through the utilization of Encoding/Decoding Tables which effectively assign symbols to represent frequently encountered strings, i.e., the symbols of the original alphabet have been displaced or substituted. The displaced original symbols are represented by entries in one of the Secondary Encoding/Decoding Tables.

The Substitution Method of compressing data contained within files uses the structure of the language and the probability of symbols and strings occurrence. Thompson investigated four classes of coding automata: character encoding, word encoding, grammar encoding, and parse encoding [87]. The Substitution Method combines features of character and word encoding. Entries in the Encoding/Decoding Tables contain characters (symbols) of the language, strings enumerated in the Source Language Syntax

Table, repeated symbol strings, and entries generated from string syntax which were contained in the Source Language Syntax Table. String syntax is expressed in the Source Language Syntax Table along with a positional attribute, if applicable. Nomenclature and equations are developed for generating and counting strings satisfying the indicated string syntax, enumerated strings, symbol occurrences, and repeated character strings during examination of files of a particular source language.

Symbol occurrences are utilized to calculate entropy, H[c0], and conditional entropy, H[c1], based upon knowledge of one preceding symbol. Each string is assigned a weight calculated by assuming that the string will be represented by only one character instead of the number of characters in its original length. Each repeated character string is assigned a weight based on the difference between representation using one reassigned symbol and representation by a repeat symbol, <repeat>; multiplier symbol(s), <count>; and the assigned symbol, <symbol>. The original symbols are assigned weights based on the number of times that each symbol is encountered. Encoding/Decoding Tables are developed using the weighting factors of the respective strings enumerated or generated from entries in the Source Language Syntax Table, repeated character strings, and the original symbols. The Encoding/Decoding

Tables effectively map a variable-length source block into a variable-length code word with each code word being a multiple of normal character size.

SUBSTITUTION METHOD DESCRIPTION

The Substitution Method of encoding/decoding files requires that a group of files be provided which contain records (sentences) written in the language for which compression is desired. In addition, a Source Language Syntax Table (SLST) is required which contains entries of string syntax or strings themselves which should be utilized by someone creating a file in the respective source language. The set of files and the Source Language Syntax Table serve as inputs to the Redundancy Analysis Program (RAP). Figure 5 shows the flow of information in the development of the Substitution Method of encoding/decoding.

Source Language Syntax Table

The Source Language Syntax Table contains syntactic strings and/or strings themselves. These entries may contain trailing blanks by simply including the number of desired trailing blanks in the string length, |n| . The Redundancy Analysis Program does not suppress string trailing blanks if they have been explicitly indicated in the Source Language Syntax Table as being required to match a specified string. Repeated character strings do not have to be specified

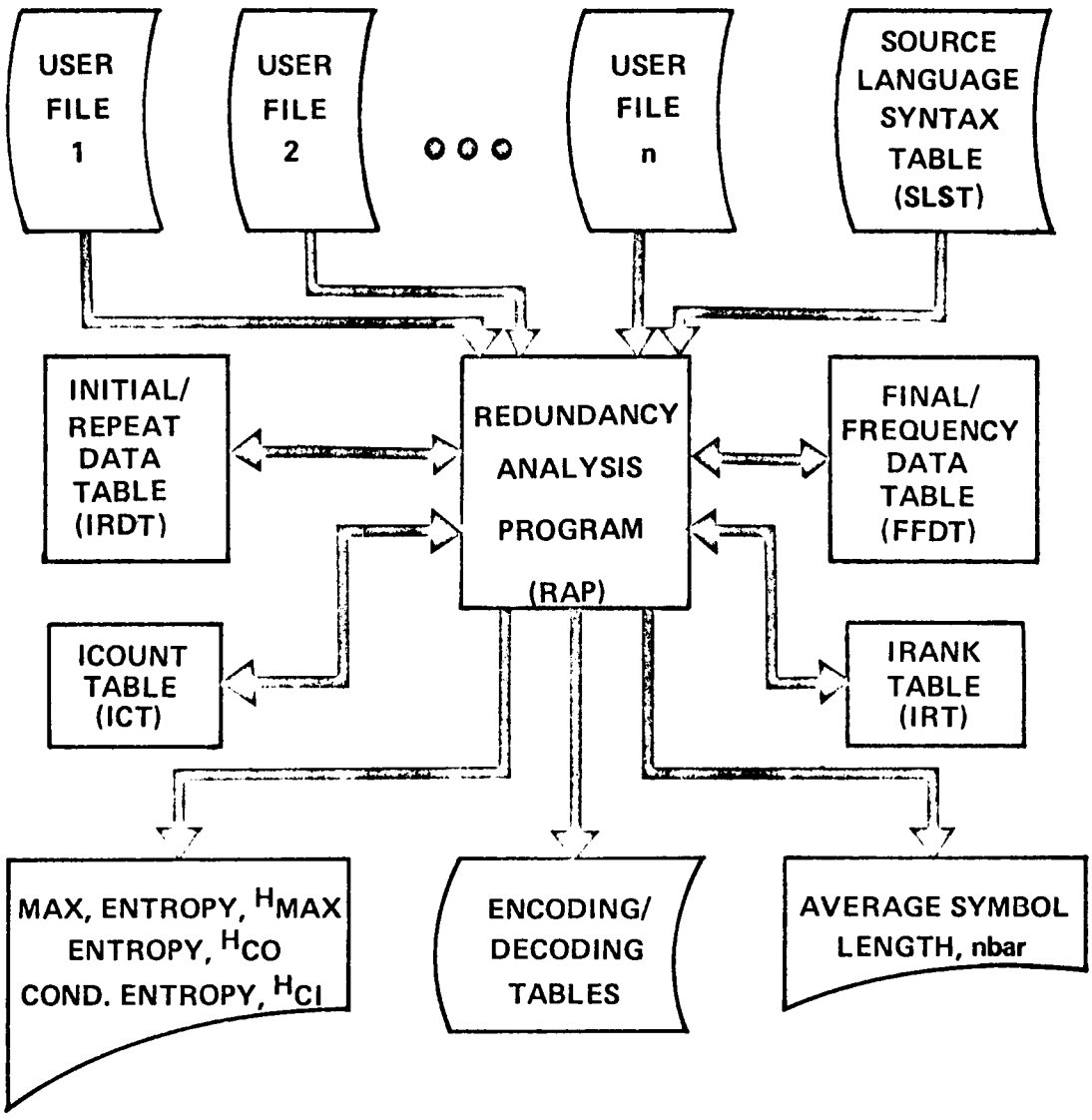

FIGURE 5. DEVELOPMENT OF SUBSTITUTION ENCODING/DECODING METHOD

because the program automatically counts such strings, and if they occur frequently enough, then they are considered potential candidates for entry into one of the Encoding/Decoding Tables.

Entries in the Source Language Syntax Table consist of five (5) fields. The first four fields are generated by the user with the fifth field calculated within the program. The first field indicates whether the entry is a syntactic entry or string itself. A syntactic entry contains pound signs in string positions for which an entry is to be generated if a string is contained within the files which matches the characters not represented by pound signs. The second field indicates whether a syntactic entry or string has a positional attribute within the record which must also be considered. The third field indicates the length of the syntactic entry or string. The length must be equal to or less than 24 characters. The fourth entry contains the syntactic entry or string itself. The fifth field is used to indicate the number of strings generated by a syntactic entry or the number of times that a delineated string is encountered during examination of the respective source language files.

Entries in the Source Language Syntax Table become the first entries in the ICOUNT Table which has a similar format and is described below.

Final/Frequency Data Table

The Final/Frequency Data Table (FFDT) is used to count the number of times each symbol of the language is encountered during examination of each record of every file in the set of files for the language being examined. The FFD Table has entries which contain the following information:

1. The number of times that any symbol, i, is followed by any other symbol, j, anywhere within the record.
2. The number of times any symbol, i, appears as the last character within a record.
3. the sum of times each symbol, j, follows all other symbols.
4. The sum of times each and every symbol, i, was encountered.
5. The total number of characters contained within every examined record.

The FFD Table allows calculation of maximum entropy, H[max]; entropy, H[c0]; and conditional entropy, H[c1]. Entropy and conditional entropy provide indicators of the maximum data compression which can be obtained using fixed-to-variable length encoding.

Since records of BCD files are constrained to be of a certain fixed length, a count is maintained of the number of records for which the last character was a blank, and for

such records, the number of trailing blanks is counted since an end-of-record character, <eor>, could be used to compress such blanks. Care has been taken to not count as excess those trailing blanks contained in strings either generated or enumerated from entries in the Source Language Syntax Table. The FFD Table does not include excess trailing blanks since an end-of-record, <eor>, symbol can be used to suppress the superfluous characters.

The information tabulated in the FFD Table is used to calculate entropy, H[c0], and conditional entropy, H[c1], respectively. The required probabilities are determined as follows:

$$p[i] = \frac{\text{Row Sums}}{\text{Total Symbols}} = \frac{FFD(i,66)}{FFD(65,66)} \qquad (3\text{-}1)$$

and,

$$p[i,j] = \frac{\text{Joint Sum}}{\text{Total Symbols}} = \frac{FFD(i,j)}{FFD(65,66)} \qquad (3\text{-}2)$$

Using these probabilities and equations 2-5 and 2-6, entropy and conditional entropy can be calculated as shown below:

$$H[c0] = 3.322 \sum_{i=1}^{m} p[i] \log 10 \ (1/p[i]) \quad \text{bits/symbol} \qquad (2\text{-}5)$$

$$H[c0] = \frac{3.322}{FFD(65,66)} \sum_{i=1}^{64} FFD(i,66) \log[10] \left(\frac{FFD(65,66)}{FFD(i,66)}\right)$$

bits/symbol (3-3)

and,

$$H[c1] = 3.322 \sum_{i=1}^{m} \sum_{j=1}^{m} p[i,j] \log[10] (p[i]/p[i,j])$$

bits/symbol (2-6)

$$H[c1] = \frac{3.322}{FFD(65,66)} \sum_{i=1}^{64} \sum_{j=1}^{65} FFD(i,j) \log[10] \left(\frac{FFD(i,66)}{FFD(i,j)}\right)$$

bits/symbol (3-4)

The summation index j has one more entry to account for the times that a symbol appears as the last character in a record. Equations 3-3 and 3-4 apply specifically to BCD files.

Initial/Repeat Data Table

The Initial/Repeat Data Table tabulates the number of times each symbol appears in the first character position of a record and the number of times each symbol consecutively repeats. Symbol repeats are indicated by using a repeat character, <repeat>, e.g., <XXXXX> ::= <repeat><5><X>. The

IRD Table is constructed assuming that an end-of-record character exists to remove superfluous trailing blanks.

The IRD Table indicates repeated character strings for which it may be advantageous to represent the repeated character string by one character instead of by three or four, i.e., <XXXXX> could be represented as <repeat><5><X> (3 characters), but it may be advantageous to represent <XXXXX> by <new symbol> (1 character). Example 1 shows how the FFD and IRD Tables are used to tabulate information for a specified sequence of symbols.

ICOUNT and IRANK Tables

The ICOUNT Table is used to count strings which are found in records in the set of examined files. The first entries to be placed in the ICOUNT Table are the syntactic and delineated strings found in the Source Language Syntax Table. Strings in file records which match the syntactic entries cause additional entries to be entered into the ICOUNT Table. After examination of all records in the files, repeated strings which occur frequently, symbols in the original alphabet, an end-of-record character, and a repeat character become additional entries in the ICOUNT Table. The entries are sorted by weight in the IRANK Table. The ICOUNT Table is rebuilt using ISORT to facilitate generation of appropriate Encoding/Decoding Tables. A table-shift character is generated for each table used beyond the

Example 1. Construction of FFD and IRD Tables

Information Applicable to Following Sequence

0 1 1 0 1 1 0 0 1 0 1 0 0 0

Final/Frequency Data Table for Specified Sequence

n[0(last)] = 1	n[ij=00] = 3	n[ij=01] = 4	n[i=0] = 8
n[1(last)] = 0	n[ij=10] = 4	n[ij=11] = 2	n[i=1] = 6
n[p(last)] = 1	n[j=0] = 7	n[j=i] = 6	Σn[i] = 14

Initial/Repeat Data Table for Specified Sequence

n[0(first)] = 1	n[0↑2] = 1	n[0↑3] = 1	... = 0
n[1(first)] = 0	n[1↑2] = 2	n[1↑3] = 0	... = 0

Primary Encoding/Decoding Table.

SUBSTITUTION METHOD ALGORITHM

The algorithm for the Substitution Method of encoding/decoding files produces effective Encoding/Decoding Tables and an estimate of its potential for compressing files by removing unwanted redundancy. Steps of the algorithm are shown in Figure 6 and enumerated below:

1. Read syntactic and fixed strings from the Source Language Syntax Table.
2. Read a filename from the file which contains the names of all files to be examined.
3. Access the file.
4. Read and reformat each record such that each character of the record (sentence) is placed in the least significant character position of a word in the character buffer.
5. Suppress trailing blanks by using a dummy end-of-record character, <eor>, and count the trailing blanks which could be removed. The dummy end-of-record character is not required if the next record begins with a specified or generated string which has

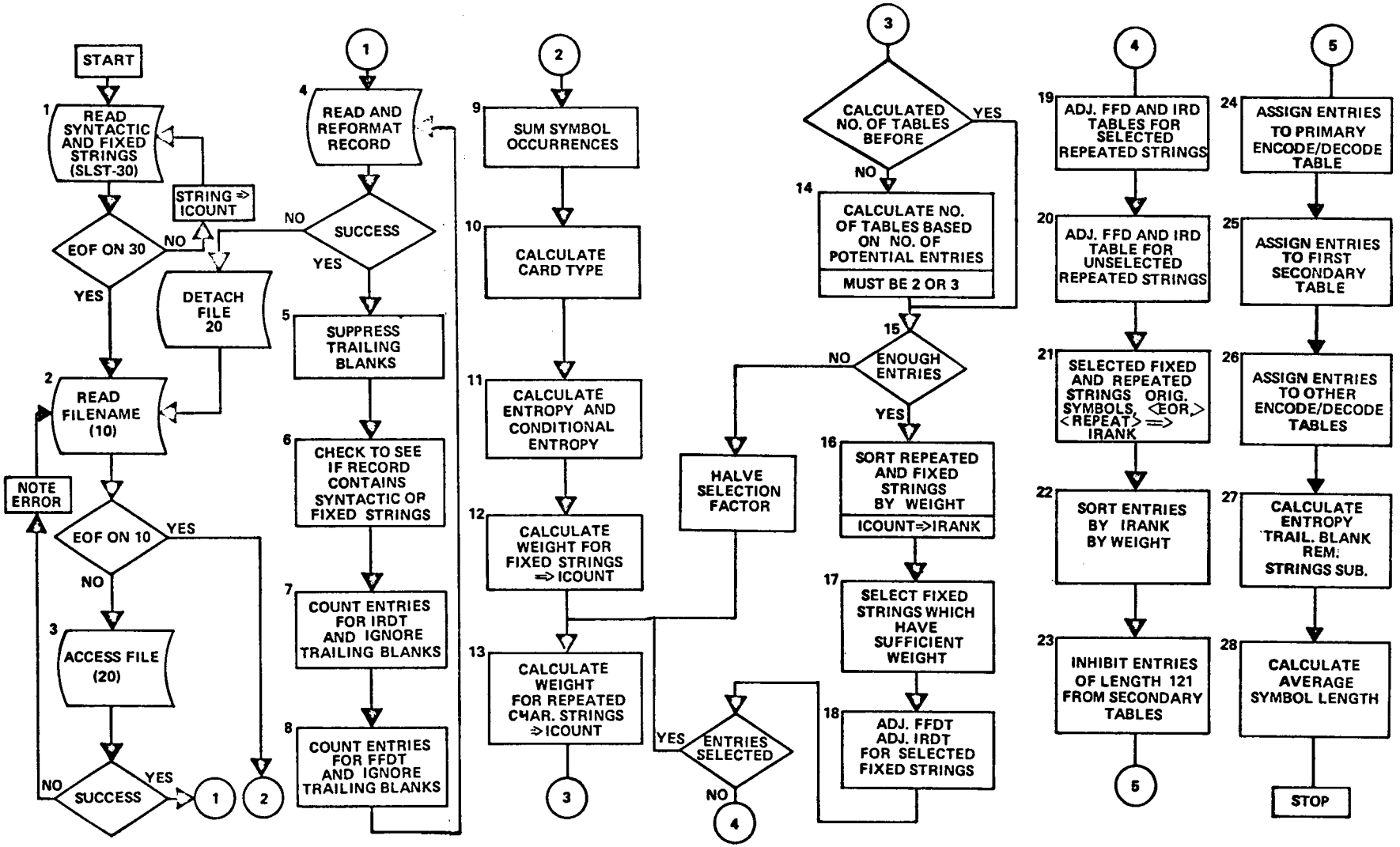

FIGURE 6. SUBSTITUTION METHOD ALGORITHM

a positional attribute of one.

6. Check to see if syntactic strings or fixed strings (from SLST) are found on this record. Strings which match syntactic strings create equivalent fixed strings. An appropriate counter is incremented each time a string match is found. Trailing blanks found in strings enumerated or generated from entries in the Source Language Syntax Table are properly accounted for.
7. Count entries for Initial/Repeat Data Table ignoring trailing blanks.
8. Count entries for Final/Frequency Data Table ignoring trailing blanks.

End processing of this record; repeat until all records processed.

End processing of this file; repeat until all files processed.

9. Sum symbol occurrences.
10. Calculate card type.
11. Calculate entropy and conditional entropy using FFD Table.

12. Calculate weights for fixed strings in ICOUNT. The string weight equals the difference in representing the string by one character rather than by the number of characters in its length multiplied by the number of times that the string was encountered.

13. Calculate the weight for repeated character strings in IRD Table. The weight equals the difference between the number of characters required to represent the string using a repeat character, <repeat><count><symbol>, rather than representation by one character, <new symbol>, multiplied by the number of times the repeated character string was encountered.

14. First time, calculate the number of tables to be used by counting the number of fixed strings and repeated character strings whose weight is greater than the number of remaining characters divided by 64 X 64. The divisor assumes the worst case, i.e., that all 64 symbols in the Primary Table point to distinct Secondary Tables. The number of tables

is presently arbitrarily restricted to three because of memory considerations.

15. Determine if there are enough entries in ICOUNT to fill the number of encoding/decoding tables to be used and also allow for syntactic entries and potential difficulty in assignment of repeated character strings of length two, |2| .

If there are not enough entries, halve the selection factor and again select entries from IRD (return to step 13); otherwise continue.

16. Sort fixed strings and repeated character strings of ICOUNT Table by weight and place in IRANK Table.
17. Select all fixed strings which exceed the weight of the last entry in IRANK necessary to fill the number of desired tables.
18. Reduce FFD and IRD Tables for symbols and repeated character strings contained in entries accepted in 17.

If fixed strings were accepted this

iteration, repeat loop (return to step 13); otherwise, continue.

19. Adjust FFD and IRD Tables to reflect the selected repeated string entries which will be represented either in the Primary of Secondary Encoding/Decoding Tables.
20. Adjust FFD Table to indicate that the unselected duplicate strings whose length is greater than three, |3| , will be represented by three or four characters instead of the number of characters, |n| , in the original string, e.g., <XXXXX> = <repeat><5><X>.
21. Fill IRANK with the selected fixed and repeated character strings; original symbols; end-of-record character, <eor>; and repeat character indicator, <repeat>.
22. Sort the selected fixed and repeated character strings, original symbols, <eor>, and <repeat> by weight. The Intermediate Statistics are stored in IRANK.
23. Suppress repeated character strings of length 2, |2| , to prevent representation in the Secondary Encoding/Decoding Tables.
24. Assign entries to the Primary Encoding/Decoding Table, Table 1. Entries in IRANK which represent

the original symbols and which occur frequently enough to be represented by one character are mapped such that the original and encoded symbol are identical, e.g., original <X> = encoded <X>. Table 1 is completed using the remaining entries in IRANK which are ranked from 1 to the number of table entries - the number of tables + 1. A table shift symbol, <table shift n>, is included in the Primary Encoding/Decoding Table for each Secondary Encoding/Decoding Table.

25. Assign entries to the first Secondary Encoding/Decoding Table, Table 2. Entries in IRANK which represent the remaining original symbols are selected without regard to weight as entries in this table such that the original and encoded symbol are identical. Table 2 is completed using the highest remaining unused entries in IRANK.
26. Assign entries to the remaining Encoding/Decoding Tables, Tables 3-n. Entries are composed of the highest remaining unused entries in IRANK.
27. Calculate entropy with blanks removed and strings substituted.
28. Calculate average number of bits required to

represent each symbol.

MATHEMATICAL BASIS

The Substitution Method removes unwanted informational redundancy through the utilization of Encoding/Decoding Tables whose entries represent the original symbols in the alphabet, character strings, repeated symbol strings, and special symbols such as <eor>, <table shift>, and <repeat>. The purpose of this section is to describe how the table entries are selected. It is assumed that a set of files written using the language of interest exists and may be examined.

All symbols of the original alphabet must be represented in one of the Encoding/Decoding Tables to not restrict usage of the language. Character strings must be counted before a decision can be made on whether to represent the string in one of the Encoding/Decoding Tables. As the alphabet size becomes large, the counting of all possible strings becomes prohibitive, e.g., for BCD (64 symbols) and allowing the string size to be between 2 and 24 symbols would require the sum of 23 factors:

$$\text{Count Locations} = \sum_{n=2}^{24} 64^n \qquad (3\text{-}1)$$

Furthermore, if one counted all strings within the files and selected frequently encountered strings on weight (based on count), one is not assured that the strings represent the language syntax, i.e., the files may not be typical. By allowing the ability to specify strings which a person writing a source file would normally use, one is assured that a finite table area is required to count the strings, and the counted strings are based on the language syntax.

Trailing blanks do not contribute to the informational content. An end-of-record, <eor>, symbol is used to indicate that there are no additional informational symbols within the record. The <eor> symbol itself may be suppressed should the first string of the next record have a positional attribute of 1, e.g.

<alphabet><alphabet>...<alphabet>(blank)...(blank)
1(string)n <alphabet>...

could be replaced by:

<alphabet>...<alphabet>1(string)n <alphabet>...

which would result in all of the blank entries of the first record simply being ignored during the counting of repeated symbols, symbol occurrences, and generated and delineated strings except where the string requires trailing blanks.

After all symbols and strings have been counted, an entry weight is calculated based on the number of symbols in

the string which would not be required should the string be represented by one new symbol in the Primary Encoding/Decoding Table. The entry's weight is based upon the number of times that the string was encountered and a factor which is a function of the number of symbols that would be displaced. Examples of weight calculation are shown below:

Fixed or Generated Strings

Original Length	Times Encountered	Entry Weight	Total Weight
|2|	10	1	10
|4|	3	3	9
|8|	6	7	42

Repeated Symbols String

Original Length	Times Encountered	Entry Weight	Total Weight
|2|	20	1	20
|3|	6	2	12
|4|	8	2	16
...			
|70|	2	3	6

Fixed and repeated string entries are sorted by weight. Fixed strings are immediately selected to be represented in one of the Encoding/Decoding Tables if its weight is equal to or greater than the weight of the last entry required to complete the specified number of Encoding/Decoding Tables, i.e.,

Number of Table Entries = 63 * (No. of Tables - 1) - 1

The minimum weight is based on the weight of the entry which occupies the last required location of the sorted table of fixed and repeated string entries. Because the fixed strings may contain repeated strings, the symbol frequency tables, FFDT and IRDT, must be adjusted, if applicable.

The process of sorting the fixed and repeated strings continues until no fixed entry is selected during the selection phase of an iteration.

When no fixed entries are selected during an iteration, the symbol frequency tables, FFDT and IRDT, are adjusted for both the selected and unselected repeated strings as shown below:

Selected Repeated Strings

⟨symbol⟩n => ⟨new symbol⟩

Unselected Repeated Strings

⟨symbol⟩n => ⟨repeat⟩⟨n⟩⟨symbol⟩ $4 \leq n \leq 63$

⟨symbol⟩n = ⟨repeat⟩(0)⟨n-64⟩⟨symbol⟩ $n \geq 64$

Unselected repeated symbol strings whose length is |2| and |3| are represented as ⟨symbol⟩⟨symbol⟩ and ⟨symbol⟩⟨symbol⟩⟨symbol⟩, respectively. All selected fixed and repeated strings, the original alphabet, and special symbols (⟨eor⟩, ⟨repeat⟩, etc.) are sorted by weight. The entries whose weight causes them to be ranked higher than 64 less the number of Encoding/Decoding Tables are selected for inclusion in the Primary Encoding/Decoding

Table. The last entries in the Primary Encoding/Decoding Tables are reserved for table shift symbols. The maximum number of tables allowed at present is 3 due to memory considerations. Repeated string entries whose original length was equal to 2, |2| , are inhibited from inclusion in Secondary Encoding/Decoding Tables since there would be no advantage, i.e., <table shift><new symbol> could be just as effectively represented as (symbol)(symbol).

Because the Substitution Method is based on a variable-to-variable length encoding method, there are no known theoretical limits to which its effectiveness can be compared. The Substitution Method's effectiveness is compared to fixed-to-variable length encoding methods in Chapter 4.

Chapter 4

RESULTS OF THE ANALYSIS OF SOURCE FILES

Two types of source files are considered in this investigation: Job Control Language (JCL) and Fortran. In addition, a technique which allows the Substitution Method to be utilized to compress data files is described. To gain insight into the effectiveness of the various encoding methods, JCL and Fortran source files were examined and encoded by both the Shannon-Fano and Substitution Methods. Comparison of the two methods demonstrates that the Substitution Method is effective and easily implemented without need for a large translation table.

Communication of file information between intelligent hosts requires only transmission of the informational symbols; but, efficient disk storage utilization requires that consideration be given to both informational and structural symbols. The Substitution Method allows simultaneous reduction of both symbol and structural redundancy.

ANALYSIS OF JCL SOURCE FILES

JCL source files were examined to determine the symbol frequency distribution which allows calculation of

entropy and conditional entropy, and also utilized to construct a Shannon-Fano code for two cases: original symbol distribution and symbol distribution with trailing blanks removed. The JCL source files were further examined to determine appropriate Encoding/Decoding Tables with entries in the Source Language Syntax Table serving as inputs to the Redundancy Analysis Program.

JCL Source File Informational Format

Control cards on the Honeywell 6080 computer are constrained by certain rules enumerated below:

1. All control cards except the ***EOF card are identified by a $ in column 1.
2. The control card name normally begins in column 8.
3. When the control card name begins in column 8, column 15 must be blank, and the variable field must begin in column 16.
4. Variables must be located between columns 16 and 72, separated by commas, equal to or less than 12 characters in length, and less than six (6) variables per card.

The Newport News Computer Center has implemented a feature called Extract JCL. Extract JCL provides the capability of selecting portions of the entire JCL based on specified bit settings found on a % JOBNAME card. Extract

JCL cards are constrained by certain rules delineated below:

1. All Extract JCL cards are identified by a % in column 1.
2. The % control card name must begin in column 8 except for % label cards which are located between columns 3 and 13.
3. Column 15 must be blank with the variable name beginning in column 16.
4. Variables must be located between columns 16 and 72, separated by commas.
5. Extract JCL requires that there be at least one (1) % JOBNAME card in the card file.

Comment and data cards are also found within production JCL. Comment cards are indicated by an asterick in column 1.

All JCL files are linked BCD files. Each 36-bit word contains six 6-bit byte characters; therefore, the alphabet consists of 2**6 or 64 symbols as shown in Appendix A.

A syntactic specification of the JCL language for the Honeywell 6080 is described below:

```
<JCL Information>      ::=   <JCL Record>*...*<JCL Record>

<JCL Record>           ::=   <Control Card Image>|
                             <Extract Card Image>|
                             <Comment Card Image>|
                             <Data Card Image>

<Control Card Image>   ::=   1($)1 *
                             2(blank)7 *
```

```
                                  8<Control Card Name>14 *
                                  15(blank)15 *
                                  16<JCL Variables>72

<Control Card Name>      ::=      ABORT | ALGOL | ... | 355SIM

<JCL Variables>          ::=      <JCL Variable>[*(,)*<JCL Variable>]5

<JCL Variable>           ::=     <Alphabet>12

<Extract Card Image>     ::=     <Extract Control Card>|
                                 <Extract Label>

<Extract Control Card>:::=       1(%)1 *
                                 2(blank)7 *
                                 8<Extract Card Name>14 *
                                 15(blank)15 *
                                 16<Extract Variables>72

<Extract Label Card>     ::=     1(%)1 *
                                 2(blank)2 *
                                 3<Extract Label>13

<Extract Card Name>      ::=     (JOBNAME) | ... | INFO

<Extract Variables>      ::=     <Extract Variable>[*(,)*<Extract
                                   Variable>]

<Extract Variable>       ::=     <Extract Label> | <Alphabet>


<Extract Label>          ::=     <Alphabet>6


<Comment Card Image>     ::=     1(*)1 *
                                 2<Alphabet>80

<Data Card Image>        ::=     1<Alphabet>80

<Alphabet>               ::=     <Letters> |
                                 <Numbers> |
                                 <Symbols>

<Letters>                ::=      A | B | ... | Z

<Numbers>                ::=      0 | 1 | ... | 9

<Symbols>                ::=      [ | * | ... | ¢
```

JCL Source File Structural Format

JCL files contain additional redundancy because of their structured nature, i.e., the presence of control/fill characters. Table I shows the structure of a BCD file on a Honeywell 6080 computer operating under GCOS SR/G. Table I demonstrates four significant attributes of BCD JCL source files which must be considered by effective data compression techniques:

1. 87.5% of each llink (320 words) is used for informational symbols.
2. Each informational record consists of fourteen words regardless of the number of required words. An end-of-record, <eor>, symbol would allow significant compression by allowing superfluous trailing blanks to be discarded.
3. 12.5% of each llink is used for control/fill characters.
4. Only 2.5% of the control/fill characters are required.

Preliminary JCL Source File Analysis

Preliminary analysis of JCL source files was accomplished with an early version of the Redundancy Analysis Program called YJCLR which only counted the number of characters found in the examined files without considering character strings. The symbol frequency

Table I. Format of BCD JCL Files

Word Number	Description	Character Usage Control/Fill Need	Excess	Data
1	Llink No.;Wds. Used	4	2	
2	Control Word 1	2	4	
3	Record1,Word 1			6
...	Words 2-13			72
16	Record 1,Word 14		4	2
17	Control Word 2		6	
18	Record 2,Word 1			6
...	Words 2-13			72
31	Record 2, Word 14		4	2
...	Records 3-20		180	1440
302	Control Word 21		6	
303	Record 21, Word 1			6
...	Words 2-13			72
316	Record 21, Word 14		4	2
317	Excess Word		6	
318	Excess Word		6	
319	Excess Word		6	
320	Excess Word		6	
		6	234	1680

n(ncf) = 6

n(cf) = 6 + 234 = 240

n(d) = 1,680

n(t) = 1,920

distribution was used to calculate entropy and conditional entropy, and to construct a Shannon-Fano code.

Execution of the YJCLR program provided the results shown in Tables II and III. Approximately 700 JCL files were examined which contained 45,031 card images and 3,602,480 symbols. Table II lists the absolute count of occurrences of each symbol, and Table III enumerates the probability of occurrence for the 15 most prominent symbols. Using the data contained within the FFD Table, H[c0] and H[c1] were calculated to be 1.8155 and 1.1529 bits/symbol, respectively. The Shannon-Fano technique was used to determine symbol representation based on the probability of occurrence [2,69]. An approximate value for the average symbol length, nbar, was found to be 2.392 bits/symbol (calculation is shown in Table III). It is not felt that Huffman encoding would offer too much improvement because of the high probability (.734) of a blank, and a blank is already represented by one bit in the Shannon-Fano encoding.

The Shannon-Fano representation did not produce a "good" match between the source (file) and the channel, i.e., nbar > H[c0] or 2.392 > 1.8155. The major reason for the poor representation is due to the high probability of a blank; thus, an end-of-record character, <eor>, will be introduced to allow superfluous trailing blanks to be suppressed.

Table II. YJCLR Execution Results

Control Cards	37,787
Comment/Data Cards	7,244
Total Cards	45,031

0 =	48790	1 =	37534	2 =	18777	3 =	11998
4 =	10083	5 =	10192	6 =	12011	7 =	5362
8 =	4198	9 =	11562	[=	1	# =	445
@ =	10	: =	205	) =	0	? =	17
=	2644431	A =	31203	B =	14491	C =	18805
D =	36535	E =	47017	F =	18277	G =	4637
H =	6489	I =	34979	& =	56	. =	3971
] =	6	(=	269	< =	11	- =	5
↑ =	6	J =	5015	K =	8191	L =	31744
M =	17183	N =	49532	O =	24169	P =	27492
Q =	395	R =	38038	- =	12689	$ =	37967
* =	66855	) =	279	; =	127	' =	55
+ =	62	/ =	13864	S =	41534	T =	59519
U =	10546	V =	3310	W =	2889	X =	10163
Y =	3021	Z =	12259	_ =	6	, =	89076
% =	13	= =	1470	" =	2614	! =	30

SUM = 3,602,480

H(max) = 6.0000 H(c0) = 1.8155 H(c1) = 1.1529

Table III. JCL Shannon-Fano Code With Trailing Blanks (9/74)

Symbol	Occurrences	Probability	S-F Code	Length
blank	2644431	.7340	0	1
,	89076	.0247	10000	5
*	66855	.0186	10001	5
T	59519	.0165	10010	5
N	49532	.0137	10011	5
0	48790	.0135	10100	5
E	47017	.0131	10101	5
S	41534	.0115	10110	5
R	38038	.0106	101110	6
$	37967	.0105	101111	6
1	37534	.0104	110000	6
D	36535	.0101	110001	6
I	34979	.0097	110010	6
L	31744	.0088	110011	6
A	31203	.0087	110100	6
Others	307726	.0856	11------	8
Totals	3602480	1.0000		

nbar = 1*.7340 + 5*.112 + 6*.0688 + 8*.0856
= .734 + .560 + .413 + .685
= 2.392

Effectiveness = H(c0)/nbar = 1.8155/2.392 = .7589

Production JCL at the Newport News Computer Center is dynamic, i.e., files are constantly being updated, added, and deleted; but, the characteristics of symbol usage remains essentially constant. Several runs of YJCLR yielded the following data:

Date	H C0	H C1
09/20/74	1.8155	1.1529
12/19/74	1.8278	1.1704
04/04/75	1.8132	1.1579
12/31/75	1.8824	1.1857
01/07/76	1.8135	1.1774
Average	1.8304	1.1688

Thus, the assumption that the source is ergodic appears to be valid based upon the above data.

JCL Source File Analysis Using RAP

The Redundancy Analysis Program, RAP, implements the Substitution Method Algorithm described in Chapter 3. Encoding/Decoding Tables were generated based upon entries found in the Source Language Syntax Table. Results of the analysis of JCL source files are found in Appendix C.

JCL Source Language Syntax Table

The JCL Source Language Syntax Table consists of two syntactic entries and four fixed entries which represent strings commonly used on tape and disk cards at the Newport News Computer Center. Syntactic entries can be readily

identified by two attributes: they are identified by a 1 in the first field of the entry and the entry itself contains pound sign(s), #'s. The first syntatic entry, <$ ####### >, causes strings to be generated and counted which match the format of the portion of the control card which indicates the control card type. The positional attribute of 1 associated with this syntactic entry and the entry itself signify the following:

1($)1 *

2(blank)7 *

8<#######>14 *

15(blank)15

Thus, all card types found within the production JCL are automatically generated by this one syntactic entry.

The second syntactic entry, <% ####### >, causes strings to be generated and counted which match the format of the portion of the control card which indicates the Extract card type. The positional attribute of 1 associated with this syntactic entry and the entry itself signify the following:

1(%)1 *

2(blank)7 *

8<#######>14 *

15(blank)15

Thus, entries representing all Extract card types found

within the examined files are automatically generated.

The last four (4) entries in the Source Language Syntax Table are used to identify fixed strings which should be utilized by someone writing a JCL at the Newport News Computer Center. All four fixed strings refer to tape and file descriptive terms and have no positional attribute.

JCL Source File Structural Redundancy

Table IV provides a summary of JCL source file statistics. The first column identifies the characteristics of the file as they presently exist. The most significant feature is that structural characters utilize 65% of all characters within the files. Trailing blanks are the major reason for this undesirable characteristic. In addition, all characters in each record's control word are superfluous since each record consists of fourteen (14) informational words. The last informational word in each record contains four (4) fill characters. The last four (4) words in each completely utilized llink (320 words) also contains fill characters. Encoding methods described in the literature do not address the problem of compaction of structural characters; but, structural characters may make a major impact upon storage considerations.

JCL Source File Representation with Trailing Blanks

The average entropy and conditional entropy for JCL files with trailing blanks was shown to be 1.8304 and 1.1688

Table IV. Summary of JCL Source File Statistics

Number of Examined Files: 99
Number of Records: 6,604
Number of Original Symbols: 528,320

Structural Characters	BCD Sequential	Blankless BCD Sequential	Substitution Encoding BCD/RANDOM
Control			
Necessary	942	472	450
Unnecessary	41,160	40,650	0
Fill	31,576	17,590	0
Trailing Blanks	318,799	0	0
Sub-Total	392,447	58,712	450
Informational Characters			
Sequence	0	0	0
Symbols	209,521	216,125	103,647
Sub-Total	209,521	216,125	103,647
Total Characters	601,998	274,837	104,097
Total Words	100,333	45,807	17,350
Required Llink's	314	144	55
H [Cl]	1.1688	2.4697	
H[C0]	1.8304	4.2510	5.3978
nbar	2.3668	4.3991	2.8774
Informational Representation			
Based on H[cl]	102,917	88,961	
Based on h[c0]	161,173	153,125	93,244
Based on nbar	208,405	158,593	103,647

bits/symbol, respectively. Using the Input File Statistics found in Appendix C, The probabilities of the 16 most prominent symbols were calculated and delineated in Table V. These probabilities were utilized to generate a Shannon-Fano code which has an average symbol length, nbar, of 2.3668 bits/symbol. Thus, the nbar, 2.3668, obtained from examining 99 JCL source files almost equals the nbar, 2.392, obtained by examining about 700 JCL source files about 18 months earlier.

File statistics indicate that there were 528,320 informational symbols in the original records. The number of required symbols can be calculated by multiplying the total number of original symbols by nbar or H[cn] and then dividing the product by H[max], e.g., 528,320 * 2.3668 / 6.0 = 208,405. Using the specified Shannon-Fano code would require 208,405 symbols while the theoretical limit for symbol-by-symbol fixed-to-variable length encoding is 161,173. Two-character block-by-block fixed-to-variable length encoding would theoretically require only 102,917 symbols to represent the original informational symbols.

The previously described files were further analyzed to ascertain the advantages of introducing an end-of-record symbol, <eor>, which would allow the 318,799 superfluous trailing blanks to be removed. The <eor> symbol provides a further structural character savings because of the reduced

Table V. JCL Shannon-Fano Code With Trailing Blanks (3/76)

Symbol	Occurrences	Probability	S-F Code	Length
blank	388,781	.7359	0	1
,	12,208	.0231	10000	5
*	9,771	.0185	10001	5
T	8,909	.0169	10010	5
N	7,507	.0142	10011	5
E	6,871	.0130	10100	5
0	6,803	.0129	10101	5
S	6,520	.0123	10110	5
I	5,549	.0105	101110	6
D	5,395	.0102	101111	6
$	5,387	.0102	110000	6
R	5,021	.0095	110001	6
I	4,830	.0091	110010	6
A	4,632	.0088	110011	6
L	4,487	.0085	110100	6
O	4,099	.0078	110101	6
Others	41,550	.0786	11------	8
Totals	528,320	1.0000		

nbar = 1*.7359 + 5*.1109 + 6*.0746 + 8*.0786
= .7359 + .5545 + .4476 + .6288
= 2.3668

Effectiveness = H[C0]/nbar = 1.8304/2.3668= .773

number of llink's required to contain the structural and informational symbols. The number of <eor> characters used are included in the informational characters total to facilitate a later comparison with the Substitution Method.

JCL Source File Representation With Trailing Blanks Removed

Entropy and conditional entropy for JCL files with trailing blanks removed are found to be 4.2510 and 2.4697 bits/symbol, respectively. The Shannon-Fano code shown in Table VI leads to an nbar of approximately 4.4029 bits/symbol. Approximately 158,459 characters are required to represent the informational characters using the specified Shannon-Fano code. The theoretical minimum based on H[c0] is 153,125 characters. The minimum representation using two-character block-by-block fixed-to-variable length encoding is 88,961 characters.

Using the previously described entries in the Source Language Syntax Table and the data in the examined JCL files, execution of the Redundancy Analysis Program generated two (2) Encoding/Decoding Tables which contain 127 selected symbols or strings. All symbols of the original alphabet are represented regardless of the number of times that the symbol was encountered. A listing of the 127 selected entries, sorted by weight, is included in the JCL Intermediate Statistics section of Appendix C. The two selected Encoding/Decoding Tables follow the Intermediate

Table VI. JCL Shannon-Fano Code Without Trailing Blanks

Symbol	Occurrences	Probability	S-F Code	Length
blank	69,982	.3241	00	2
,	12,208	.0565	0100	4
*	9,771	.0452	0101	4
T	8,909	.0412	0110	4
N	7,507	.0347	0111	4
E	6,871	.0318	1000	4
0	6,803	.0314	1001	4
EOR	6,604	.0305	1000	4
S	6,520	.0301	1001	4
I	5,549	.0256	10100	5
D	5,395	.0249	10101	5
$	5,387	.0249	10110	5
R	5,021	.0232	10111	5
I	4,830	.0223	11000	5
A	4,632	.0214	110010	6
L	4,487	.0207	110011	6
Others	45,649	.2115	11------	8
	216,125	1.0000		

nbar = 2*.3241 + 4*.3014 + 5*.1209 + 6*.0421 + 8*.2115
= .6482 + 1.2056 + .6045 + .2526 + 1.6920
= 4.4029

Effectiveness = H[c0]/nbar = 4.2510/4.4029 = .966

Statistics section. The selected assignment causes the frequency of symbol occurrences to change to those enumerated in Output Statistics - Blanks Removed and Strings Substituted. The most frequently occurring symbol is now a "!" with a probability of .0688. The least frequently observed symbol is the "_" with a probability of .00054. Thus, a "!" is observed 127 times for every observation of a "_". The Primary Encoding/Decoding Table contains the most frequently encountered strings and symbols. In addition, a table shift symbol, <table shift 2>, is utilized to indicate that the following symbol is not to be decoded from the Primary Encoding/Decoding Table, but from a Secondary Encoding/Decoding Table.

It requires an average of 2.8774 bits to represent each symbol when the generated and listed Encoding/Decoding Tables are used. It would require 103,647 symbols to represent the original symbols with trailing blanks removed.

Further compression is possible if the output generated using the Substitution Method were further encoded using fixed-to-variable length encoding methods. The theoretical limit based on H [c0], 5.3978, would require a minimum of 93,244 symbols.

Structural characters for the BCD, sequential representation of JCL source files with trailing blanks suppressed comprise 21.4% of the total required characters.

Structural characters for the BCD, random representation using the Substitution Method comprise only .4% of the total required number of characters.

The number of structural and informational characters required to represent JCL source files using the Shannon-Fano and the Substitution Methods are shown graphically in Figure 7.

ANALYSIS OF FORTRAN SOURCE FILES

Fortran source files are generally written utilizing the ASCII alphabet. Each of the 128 ASCII characters can be represented by 7-bits. An ASCII character representation requires 9-bits on the Honeywell 6080 because of its 36-bit word length, i.e., 36 can be divided by 9 an even number of times. The Fortran character set is a subset of the ASCII alphabet. Although lower-case letters are acceptable to Fortran (mapped into upper-case letters at compile time), many currently used terminals do not have the ability to utilize this feature. Thus, all Fortran source files were converted from ASCII to BCD format with trailing blanks suppressed before analysis to be able to utilize the Redundancy Analysis Program developed to analyze JCL source files. Of course, if the Substitution Method were implemented on a production basis, then all allowed input symbols would have to have a representation.

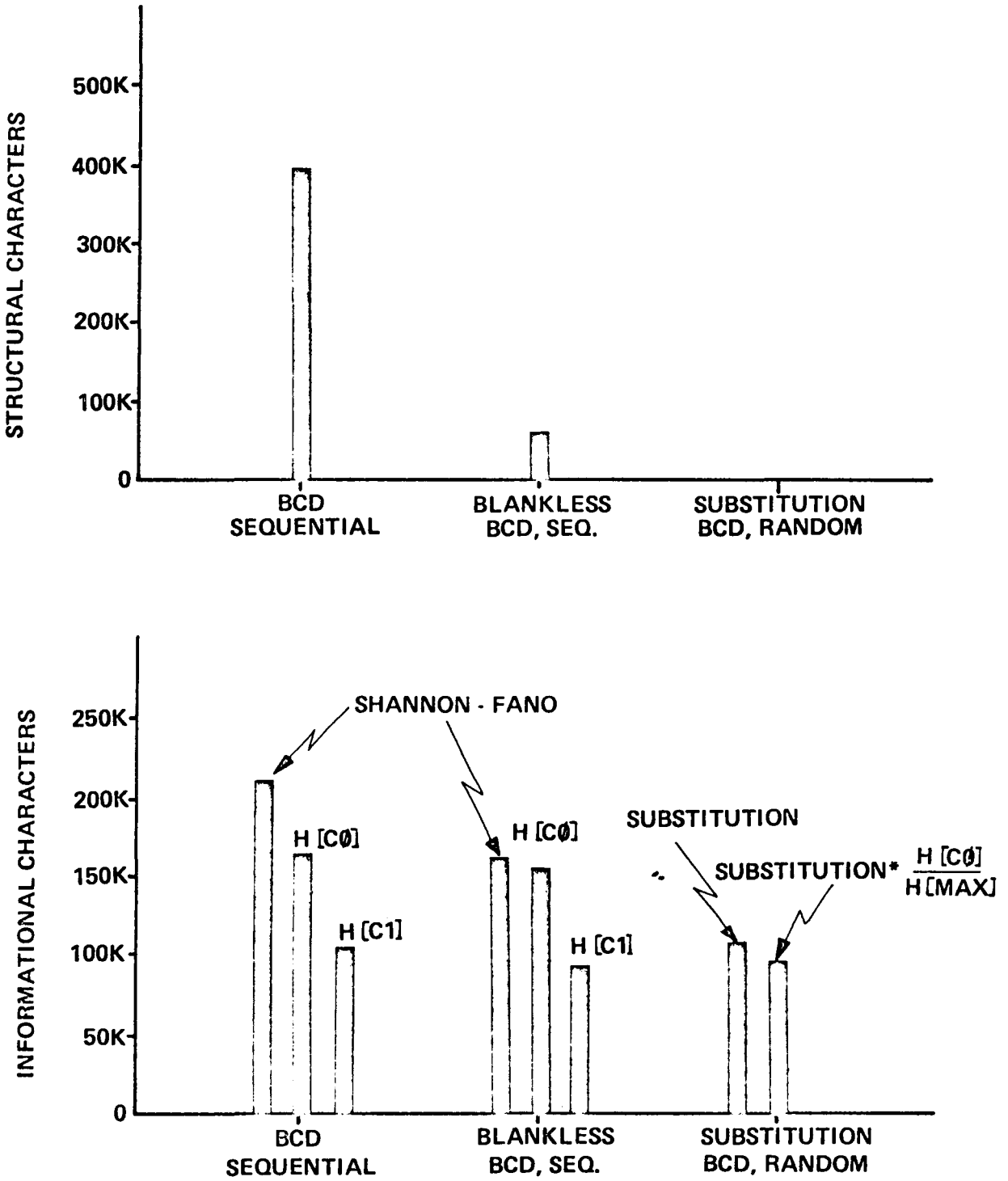

FIGURE 7. COMPARISON OF METHODS FOR COMPRESSING JCL SOURCE FILES

Fortran source files were examined to determine the symbol frequency distribution which allows calculation of entropy and conditional entropy. A Shannon-Fano code was generated based on the symbol frequency distribution. The Fortran source files were further examined to determine appropriate Encoding/Decoding Tables.

Fortran Source File Informational Format

Each record of a Fortran source file is composed of two fields. The first field is the sequence number. The sequence number consists of from one (1) to eight (8) numeric characters. The sequence field is terminated by any non-numeric character. The second field describes either a statement which the Fortran compiler is to interpret or a comment which the reader of the source program listing is to interpret.

A syntactic specification of Fortran source files is shown below:

<FORT Information>	::=	<FORT Record>*...*<FORT Record>
<FORT Record>	::=	<FORT Statement>| <FORT Comment>
<FORT Statement>	::=	<Seq. Number> * <Statement>
<FORT Comment>	::=	<Seq. Number> * (C | *) * <Alphabet>
<Seq. Number>	::=	1<Numbers>1 [*<Numbers>]7
<Statement>	::=	<FORT Commands> */ <Symbolic Names> */

```
                                <Syntax Characters> */
                                <Literals> */
                                (Blanks)

<FORT Commands>        ::=      CALL | DO | ... | WRITE

<Symbolic Names>       ::=      <Letters>[*<Alphanumeric>]7

<Syntax Characters>    ::=      " | $ | ... | *

<Literals>             ::=      <Alphabet>

<Alphabet>             ::=      <Letters> |
                                <Numbers> |
                                <Symbols>

<Alphanumeric>         ::=      <Letters> | <Numbers>

<Letters>              ::=      A | B | ... | Z
                                a | b | ... | z

<Numbers>              ::=      0 | 1 | ... | 9

<Symbols>              ::=      [ | * | ... | !
```

All Fortran source file records contain statement or sequence numbers which provide the ability to readily change the file as desired. On the Honeywell 6080, sequence numbers comprise the first part of each record. The Honeywell computer provides the capability of automatic sequence number generation with each statement number monotonically increasing by a specified delta. In addition, the capability to resequence a file as desired is available. Thus, a new symbol is introduced, <del>, which conveys the following meaning:

1. End Previous Record.
2. Add Delta to Former Sequence Number.

Thus, the <del> character has the potential of saving from 1

to 8 characters/record in addition to previously suppressed trailing blanks. The <del> character was substituted before any attempt at further data compression utilizing either the Shannon-Fano or Substitution Method of encoding.

Fortran Source File Structural Format

Fortran ASCII source files contain additional redundancy because of the presence of control/fill characters. Structural characters composed 16% of all characters within the examined files. Each file contains a twenty (20) word record in the first llink which serves as a place for control information; but at present, this area is essentially unused. Each record control word serves the same function that one end-of-record symbol could perform. Since the control word for the next record follows the last word of the previous record, fill characters are almost always required to complete a record.

Shannon-Fano Fortran Source File Representation

Using the Input File Statistics found in Appendix C, the probabilities of the 16 most prominent symbols were calculated and listed in Table VII. A Shannon-Fano code was generated based on the respective symbol probabilities. The average symbol length, nbar, was calculated to be 4.6929 bits/symbol. Using the specified Shannon-Fano code, it would require 48,246 symbols to represent the informational characters. The theoretical minumum required number of

Table VII. Shannon-Fano Code for Fortran Source Files

Symbol	Occurrences	Probability	S-E Code	Length
blank	19,833	.3215	00	2
,	3,123	.0506	0100	4
I	3,080	.0499	0101	4
T	2,721	.0441	0110	4
DEL/EOR	1,951	.0316	0111	4
C	1,787	.0290	10000	5
O	1,666	.0270	10001	5
(	1,649	.0267	10010	5
)	1,646	.0267	10011	5
1	1,587	.0257	10100	5
A	1,553	.0252	101010	6
E	1,545	.0250	101011	6
6	1,392	.0226	101100	6
N	1,276	.0207	101110	6
D	1,260	.0204	101111	6
0	1,237	.0201	110000	6
Others	14,378	.2332	11------	8
Totals	61,684	1.0000		

nbar = 2*.3215 + 4*.1762 + 5*.1351 + 6*.1340 + 8*.2332
= .6430 + .7048 + .6755 + .8040 + 1.8656
= 4.6929

Effectiveness = H[c0]/nbar = 4.3316/4.6929 = .923

symbols based on H[c0] and H[c1] are 44,532 and 35,698, respectively.

Fortran Source Language Syntax Table

The first entry in the Fortran Source Language Syntax Table indicates that the files to be examined are sequenced. This entry causes the Redundancy Analysis Program to count the number of times that an end-of-record, <eor>, symbol could be suppressed from the previous file record because of the positional attribute of sequence numbers. Further data compression may be applicable if the change in the sequence number equals the expected delta.

No syntactic entries are contained in the Fortran Source Language Syntax Table; but, there are 31 fixed entries. The first 26 fixed entries represent Fortran commands, e.g., CALL, DO, WRITE, etc. The last 5 fixed entries represent strings which one might frequently utilize with the Fortran Commands.

Fortran Source File Representation Using the Substitution Method

The Fortran source files were further analyzed using the Redundancy Analysis Program. Encoding/Decoding Tables were generated based upon entries found in the Source Language Syntax Table which were previously described. Results in the analysis of Fortran source files are found in Appendix C. The results are summarized in Table VIII.

Table VIII. Summary of Fortran Source File Statistics

Number of Examined Files: 4
Number of Records: 1,951

Structural Characters	ASCII Sequential	Syntax BCD Sequential	Substitution Encoding BCD/Random
Control			
Necessary	2,073	124	39
Unnecessary	6,311	11,846	0
Fill	4,067	5,742	0
Trailing Blanks	0	0	0
Sub-Total	12,451	17,712	39
Informational Characters			
Sequence			
Number	5,421	4	4
Delta	0	1,947	1,947
Symbols	59,729	59,733	40,302
Sub-Total	65,150	61,684	42,253
Total Characters	77,601	79,396	42,292
Total Words	19,401	13,233	7,049
Required Llink's	61	42	23
H[cl]		2.6977	
H[c0]		4.3316	5.4298
nbar		4.6929	4.1099
Informational Representation			
Based on H[cl]		35,698	
Based on H[c0]		44,532	38,238
Based on nbar		48,246	42,253

Execution of the Redundancy Analysis Program using the Source Language Syntax Table and the four (4) Fortran source files which comprise Appendix B resulted in the generation of two (2) Encoding/Decoding Tables which contained 127 selected symbols or strings. All symbols of the original BCD alphabet are represented regardless of the number of times that the symbol was encountered. A third Encoding/Decoding Table could easily be utilized to properly encode/decode characters in the original ASCII alphabet and not in the BCD alphabet.

Using the generated Encoding/Decoding Tables resulted in nbar being equal to 4.1099 bits/symbol. 42,253 symbols are required to represent the original 65,150 informational symbols after 3,466 sequence characters became superfluous with the introduction of the <del> symbol.

Figure 8 shows the number of structural and informational characters required to represent Fortran source files using the Shannon-Fano and Substitution Methods of encoding.

SUBSTITUTION METHOD UTILIZATION FOR DATA FILES

An output data file normally contains superfluous blanks in order to allow the information to be easily read and allow for larger data values to be printed without exceeding the specified field size. There are also

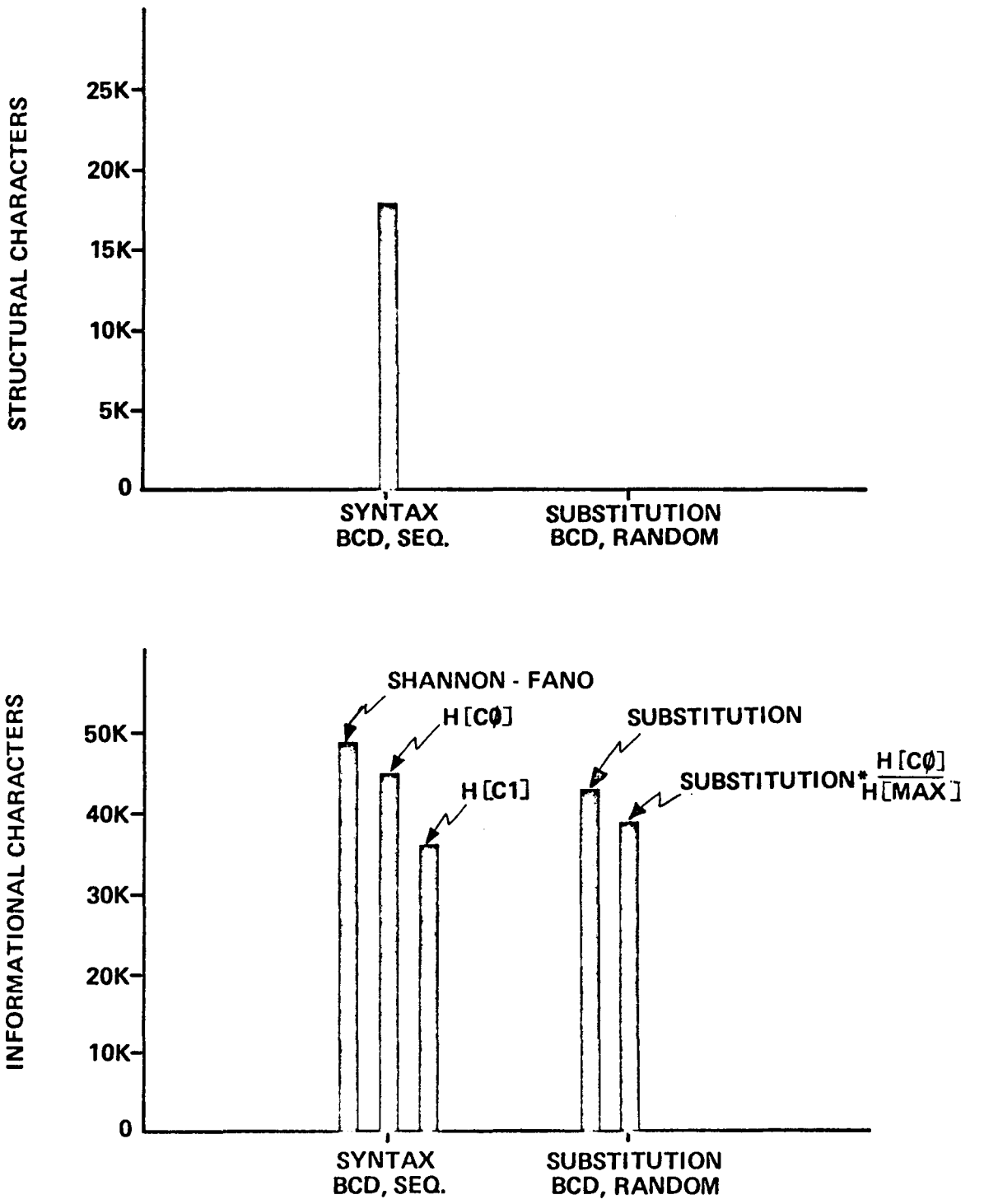

FIGURE 8. COMPARISON OF METHODS FOR COMPRESSING FORTRAN SOURCE FILES

control/fill characters which are not required from an informational point of view. A technique is described which, if implemented, would allow a program to sub-optimally compress program output as the information is being generated.

Examination of the Encoding/Decoding Tables for JCL and Fortran source files show that there are certain symbols frequently encountered in both, e.g., A, B, C, 0, 1, etc. A general Encoding/Decoding Table would be developed which contains a consensus of the prominent symbols found in earlier examinations. An encoding/decoding subroutine would be developed which provides three options as described below:

1. The least frequently anticipated general symbols would be replaced by calling and supplying the subroutine with a list of strings pertinent to a particular data file. An example of this operation is shown below:

   CALL DATAC(1,4,ITABLE)

   where ITABLE has the form:

   (ITABLE(1,J),J=1,4) = <$ FILE>

   (ITABLE(2,J),J=1,4) = <% JOBNAME>

   (ITABLE(3,J),J=1,4) = ⟨DEN16⟩

   (ITABLE(4,J),J=1,4) = ⟨NNST⟩

2. Output data would be compressed by the program's

WRITE statements being replaced with an ENCODE statement. Thus, the original data would now be formatted as before, but still in core. The Substitution Subroutine would be passed the character string which was built using the ENCODE statement. The subroutine would compress the character string and place the remaining characters in a data bucket. When the data bucket becomes filled, the subroutine would write the records to a BCD, random file. The operation can be compactly described as:

```
WRITE(10,8) A, B, C, D  =>
    ENCODE(CDATA,8) A, B, C, D
    CALL DATAC(2,10,CDATA)
```

Where, CDATA is a character string whose length equals the length of the former data record and DATAC is the name of the Substitution Method subroutine entry point.

3. Input data would be expanded by the READ statements being replaced with a call to the Substitution Subroutine and addition of a DECODE statement. The Substitution Subroutine would read the data from the previously created BCD, random file, expand the compressed record, and return it to the main program. The data would be

properly placed in the named variables by the DECODE statement. This operation can be represents as:

```
READ(10,8) A, B, C, D = >
   CALL DATAC(3,10,CDATA)
   DECODE(CDATA,8) A, B, C, D
```

Thus, data could be easily compressed and expanded with minor program modification. There would be two major advantages of such an approach:

1. Channel time would be reduced.
2. The amount of computer resources required for data storage would be reduced.

Chapter 5

ANALYSIS OF THE SUBSTITUTION METHOD USING REAL COMMUNICATION CHANNELS

The Substitution Method of encoding/decoding files has been developed in previous chapters of this dissertation. Costs of data channels and the application of the Substitution Method in a real world situation are described in this chapter. The data transmission services to be considered are:

Type of Carrier	Carrier	Service
Conventional	American Telephone and Telegraph	HiD-LoD Private Line; Dataphone Digital Service (DDS)
Specialized	Data Transmission Co., Datran	Dataline I; Dataline II
Satellite	American Satellite Corp.	Leased Channel
Value-Added	Telenet	Leased Channel

Offering of service by these principal data common-carrier companies are listed in Table IX [38,41,52,74,84,88,95]. Costs for providing one (1) data channel at 2.4 kbps and one (1) data channel at 9.6 kbps between Newport News, Virginia

Table IX. Principal Data Common-Carrier Companies

Company	Service	Status
American Satellite Corporation	American Satellite Corporation provides private voice, data, facsimile, and wideband services. Rates are significantly cheaper than those available from AT&T.	Operational.
American Telephone and Telegraph Company	Bell System provides overwhelming share of long-distance data transmission. New tariffs for WATS service and HiD-LoD data service. Dataphone Digital Service(DDS) is new offering. DDS will offer digital transmission of data at 2.4,4.8,9.6, and 56 kbps.	Operational. Available in 24 cities in November, 1975.
Data Transmission Company (Datran)	Offers digital transmission service on a private-line and switched basis. Datran provides users with transmission speeds of 2.4, 4.8, 9.6, and 56 kbps with a guarantee of 99.95 percent error-free seconds of transmission on the average.	Operational 23 cities January, 1976.
Telenet Communication Corporation	Value-added common-carrier network based on packet-switched technology developed on the Advanced Research Project Network (ARPANET). Telenet offers low, distance independent pricing, i.e., cross-country transmission at the same cost as local communication.	Operational 16 cities December, 1975.

and Houston, Texas are calculated. Tenneco's Honeywell 6080 processors are located in these two cities. The basic premise being that the 9.6 kbps channel is required to transmit data in an uncompressed form while the same information could be transmitted in compressed form over a 2.4 kbps channel if the Substitution Method could compress the data by an effective factor of four (4). An incremental cost is calculated based on a total channel usage of 175 hours/month which is slightly greater than eight (8) hours/day for five (5) days/week. Processing times and costs of encoding/decoding typical files are determined. Costs associated with pre-processing, transferring, and post-processing of the compressed file are compared with the cost of transferring the original file over facilities available from selected carrier offerings.

COMMON CARRIERS

Selection of a vendor to supply 2.4 kbps and 9.6 kbps data channels between Newport News, Virginia and Houston, Texas is not a simple matter considering the availability of offerings by common-carriers whose tariffs are based upon different factors. The picture is further complicated by the availability of common-carrier services in new cities and changing tariffs. Four types of carriers were chosen for this analysis. The analysis is based upon services currently

available and those anticipated to be available by January 1977.

1. American Telephone and Telegraph

AT&T offers data communication users private-line service based on a two-tier (High Density, HiD-Low Density, LoD) rate structure and Dataphone Digital Service (DDS). Charges for the HiD-LoD communication line are based on distance between locations for which interconnection is desired. The only fluctuation due to speed of transmission relates to the selected modems and line-conditioning. Charges for Dataphone Digital Service vary not only according to distance, but also according to speed of transmission. These two classes of services are described along with an estimate of the costs of leasing the previously described communication channels.

HiD-LoD Private Line Service

The HiD-LoD tariff establishes prices for services which are more in line with the cost of providing them than those formerly offered based on a single nationwide tariff. The premise being that the interstate network is composed of sections of concentrated (high-density) facilities which provide lower unit cost per circuit and lower-capacity (low-density) facilities which result in a higher unit cost per circuit. The tariff is composed of four charges: interexchange channels (IXC) which interconnect rate

centers, channel terminals which terminate each end of the IXC facilities, station terminals which connect IXC facilities to each station, and modems interfacing users' equipment with common-carrier facilities. Costs associated with this class of service are enumerated in Appendix D.

Dataphone Digital Service

Dataphone Digital Service offers private-line synchronous data transmission at speeds of 2.4, 4.8, 9.6, and 56 kbps. Charges vary not only according to distance, but also speed of transmission. Rates are based on charges for inter-city facilities, access lines from the digital network to the user's location, and user interface facilities. Costs applicable to this class of service are listed in Appendix D.

2. Data Transmission Company - Datran

The Datran system is designed to serve the needs of the data communication user by transmission of data in digital form in lieu of conversion from digital-to-analog form, transmission, and conversion from analog-to-digital form. Datran offers private-line and switched digital data communications in 23 cities nationwide in January 1976 with an anticipated expansion to 30 metropolitan regions by mid-1976. The Datran system provides users with transmission speeds of 2.4, 4.8, 9.6, and 56 kbps with a guarantee of 99.95 percent error-free seconds of transmission on the

average. The most outstanding feature is the Datadial which became operational in January 1975 and is capable of connecting a computer to a terminal anywhere along the Datran system in less than one second. Pricing for this service is based on a one-second minimum with a minimum billing of $.01.

Costs for 2.4 and 9.6 kbps service between Newport News, Virginia and Houston, Texas based on Datran tariffs are enumerated in Appendix D.

3. American Satellite Corporation

American Satellite Corporation (ASC) has established a nationwide network which provides private voice, data, facsimile, and wideband services. ASC has built 13 earth stations interconnected via spacing leased from the Westar satellite system. Channels are offered between ASC earth stations located in New York, Los Angeles, Dallas, Houston, Chicago, Atlanta, Pittsburg, and Washington, D. C. Costs for 2.4 and 9.6 kbps channels between Newport News, Virginia and Houston, Texas based on ASC tariffs are tabulated in Appendix D.

4. Telenet Communication Corporation

Telenet was granted authority in April 1974 by the Federal Communication Commission to establish a communication network based on packet-switching techniques which were developed on the ARPANET. Packet-switching

provides an efficient means of providing switched data services to computer users via a network of communication computers located at each Telenet Central Office (TCO) node and interconnected via terrestrial and satellite channels. Packet-switching allows the offering of distance-independent communications. Telenet implemented a value-added common-carrier service in December 1975 in sixteen cities where its TCO's are located. Users not located in these cities may access the nearest TCO using either leased or dial-up telephone lines.

Telenet's pricing structure consists of four basic charging elements: Telenet Cental Office (TCO) ports, access channels, customer-site equipment, and traffic. The charges for the Central Office port is a function of the location of the TCO, transmission speed of the port, and the amount of usage. Public-dial port charges depend on the location of the TCO and the amount of usage. Private-dial port and leased port charges depend upon the location of the TCO and transmission speed of the port. The access channel from the user to the TCO is leased by Telenet from other common-carriers and charged to the user. A Telenet Access Controller (TAC) may be installed on a user's premises to provide up to 96 asynchronous network access ports for both terminals and host access to the Telenet. Traffic through the network is priced at $0.60 per kilopacket, independent

of distance. A kilopacket consists of one-thousand packets where each packet consists of up to 1024 bits or 128 characters of user data. Costs of 2.4 and 9.6 kbps channels between Newport News, Virginia and Houston, Texas are listed in Appendix D.

LINE COSTS

The costs of providing a 2.4 kbps and 9.6 kbps channel between Newport News, Virginia and Houston, Texas are summarized in Table X. The following cost analysis is based on the premise that either of the selected data lines will be utilized 175 hours/month.

Currently, the lowest available cost for the desired communication facilities are $970/month for the 2.4 kbps line and $1,634/month for the 9.6 kbps line. The average hourly cost can be calculated by taking the monthly line cost and dividing it by the total number of hours of usage/month, e.g.,

Cost[2.4kbps,Present] = $970/175 = $5.54/hour

and,

Cost[9.6kbps,Present] = $1,634/175 = $9.34/hour

The projected charges for utilizing the communication facilities are anticipated to decrease in cost to $625/month for the 2.4 kbps line and $1,287/month for the 9.6 kbps line. The hourly costs will be:

Table X. Ranking of Offerings of Selected Carriers At 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas.

2.4 kbps

Vendor/Service	Cost	Status
Datran Dataline I Service via Norfolk	$ 625	Unavailable
AT&T Dataphone Digital Service	660	Unavailable
Datran Dataline II Service	892	Unavailable
Datran dataline I Service via Washington	970	available
Datran Dataline I Service Via Atlanta	1105	Available
ASC via Washington	1210	Available
AT&T High Density Service	1304	Available
Telenet via Washington	1454	Available

9.6 kbps

Vendor/Service	Cost	Status
Datran Dataline I Service via Norfolk	$1287	Unavailable
Datran Dataline II Service via Norfolk	1311	Unavailable
AT&T Dataphone Digital Service	1390	Unavailable
AT&T High Density Service *	1634	Available
ASC via Washington	1668	Available
Datran Dataline I Service via Atlanta	1968	Available
Datran Dataline I Service via Washington	2064	Available
Telenet via Washington	3338	Available

* - Currently Used

Cost[2.4kbps,Future] = $625/175 = $3.57/hour

and,

Cost[9.6kbps,Future] = $1,287/175 = $7.35/hour

A third alternative which will be examined is a 1.2 kbps dial-up line which has a charge of $.60/minute of usage, i.e., a dedicated line may not be justified.

PRE-PROCESSING AND POST-PROCESSING COSTS

The cost of developing Encoding/Decoding Tables must be considered in deciding whether to transfer files in their original or compressed format. Approximately $38 worth of computer resources were consumed in developing the JCL Encoding/Decoding Tables found in Appendix C.

Five (5) JCL files were chosen at random from the 99 files which were utilized to develop the JCL Encoding/Decoding Tables. The selected files were encoded using the JCL Encoding/Decoding Tables obtained from execution of the Redundancy Analysis Program. Execution of the JCL Encode Program provided the following statistics:

Filename	Input Symbols	Output Symbols	CPU Time (sec)	File I/O	Connect Time (sec)	Charge
TAR01	29,520	6,387	5.63	26	34	1.042
TAM09	8,720	2,022	1.79	9	25	.345
TBO89	9,440	2,347	2.04	9	37	.397
TBV08	11,840	2,488	2.35	11	31	.449
TDT05	7,840	1,562	1.59	7	28	.309
TOTAL	67,360	14,806	13.40	62	155	$2.542

Charge	$2.251	$.161	$.129	$2.542
%	88.6	6.3	5.07	100.0

88.6% of the pre-processing cost was due to the used processor time. Processor costs are calculated as:

Processor Cost = $8.0 * (Program Size(words)/512)
*(1/1000) * (Processor Time (seconds))

The Fortran version of the JCL Encode Program is 10,560 words.

Execution of the JCL Decode Program provided the following statistics:

Filename	Input Symbols	Output Symbols	CPU Time (sec)	File I/O	Connect Time (sec)	Charge
TAR01	6,387	29,520	6.95	27	33	$1.265
TAM09	2,022	8,720	2.15	10	26	.409
TBO89	2,347	9,440	2.04	9	31	.392
TBV08	2,488	11,840	2.70	10	28	.503
TDT05	1,562	7,840	1.79	7	24	.339
TOTAL	14,806	67,360	15.63	63	142	$2.908
		Charge	$2.626	$.164	$.118	$2.908
		%	90.3	5.6	4.1	100.0

The Fortran version of the JCL Decode Program is 10,560 words.

COMPARISON OF COSTS OF TRANSFERRING FILES IN THEIR ORIGINAL FORM VERSUS COMPRESSED FORM

The costs of transferring files in both their

original form and compressed form are shown below:

	Dial-Up 1.2 kbps		Leased	
	Original	Encoded	2.4 kbps	9.6 kbps
C[prep]		$2.542	$2.542	
C[otran]	$5.990			$ .291
C[ptran]		$1.320	$ .114	
C[post]		$2.908	$2.908	
Total	$5.990	$6.770	$5,564	$ .291

where, C[prep] equals pre-processing costs

C[otran] equals costs of transferring original file

C[ptran] equals costs of transferring compressed file

C[post] equals post-processing costs.

The costs of transferring the original and compressed files at 1.2 kbps are calculated as shown below:

$$\text{Cost (Original)} = \frac{\text{Chars. Transmitted}}{\text{Transmission Rate}} * \text{Line Charge} * \text{Line Factor}$$

$$= \frac{67{,}360}{150} * .01 * 1.33 = \$5.990$$

$$\text{Cost (Compressed)} = \frac{14{,}806}{150} * .01 * 1.33 = \$1.320$$

The 1.33 factor is based on line transmission inefficiencies such as line turn-around time.

The cost of transmission for the compressed files at 2.4 kbps and for the original files at 9.6 kbps is calculated below:

$$\text{Cost} = \frac{\text{Chars. Transmitted}}{\text{Transmission Rate}} * \text{Line Charge} * \text{Line Factor}$$

$$\text{Cost (2.4 kbps)} = \frac{14{,}806}{300} * \frac{\$5.54}{3600} * 1.5 = \$ .114$$

$$\text{Cost (9.6 kbps)} = \frac{67{,}360}{1200} * \frac{\$9.34}{3600} * 2.0 = \$ .291$$

Approximately 60% of the cost of transferring the original file could be avoided if trailing blanks could be removed, but there would be an appropriate additional charge for such processing.

The cost of transferring the original file without any data compression was less expensive than total costs for compressed transmission for both the dial-up and leased lines.

The Encode and Decode programs could be rewritten in GMAP (assembly language) to reduce their program sizes and increase their effectiveness. If the GMAP programs were 2,000 words each, and operated twice as effectively, then the following would apply:

Dial-Up

Original Costs (Blanks Removed) = .4 * $5.99 = $2.396

Compressed Costs = $.242 + $1.320 + $.277 = $1.839

Thus, there may be an advantage in compressing the files before transmission when using a dial-up line and

after the program is optimized.

Leased Line (Based on Anticipated Future Rates)

Original (No Blanks) = .4 * (7.35/9.34) * .291 = $.092

Compressed = .242 + (3.57/5.54) * .114 + .277 = $.592

Thus, there is still no advantage in compressing the files before transmission when using a leased line.

Chapter 6

CONCLUSIONS

The Substitution Method of encoding/decoding files provides an effective method of simultaneously removing unwanted symbol and structural redundancy. Symbol redundancy is removed through the utilization of Encoding/Decoding Tables which allows frequently encountered strings to be represented by only one or two symbols. Structural redundancy is removed by converting the sequential input source files to random compressed output files and through utilization of the positional attribute of entries in the Source Language Syntax Table.

The Substitution Method was used to compress two categories of source files: JCL and Fortran. JCL files consist of records whose length is fixed which results in superfluous trailing blanks. An end of record, <eor>, was used to eliminate the undesirable characteristic that 65% of the files characters were structural (control, fill, trailing blanks). The JCL files informational characters were compressed by both the Shannon-Fano and Substitution Methods. The Shannon-Fano Method resulted in 73.3% of the original symbols being retained after blank suppression while the Substitution Method retained 48%. The theoretical

minimums based on H[c0] and H[c1] were 70.9% and 41.2%, respectively. The output of the Substitution Method could be further compressed by fixed-to-variable length encoding methods to remove another 4.9% of redundancy. From a storage viewpoint, the Substitution Method would require only 17.5% of the original space to store the compressed information.

Fortran source files are sequenced on the Honeywell 6080, i.e., contain a sequence number in the beginning of each record. A new symbol, <del>, was introduced which signifies that the previous record has ended and the new record's sequence number increased by a specified delta. The <del> character was substituted before any attempt at further data compression. The Fortran source file's informational characters were compressed by both the Shannon-Fano and Substitution Methods. The Shannon-Fano code retained 78.2% of the original symbols while the theoretical limit based on H[c0] was 72.2%. The theoretical limit based on H[c1] was 45%. The substitution Method reduced the original symbols to 68.5%. The output of the Substitution Method could be further reduced to 62% if fixed-to-variable length encoding were employed. Fortran source files would require 37.7% of the original file space if compressed using the Substitution Method.

A technique was described which, if implemented, would allow a Fortran program to compress output data files,

and thus reduce channel time and storage requirements.

The Substitution Method was implemented via Fortran to allow source files to be encoded/decoded. The results show that the method is not economically feasible because of the excessive processor time costs which result when costs are calculated based on the product of core multiplied by processor time. The Substitution Method could show economic advantage if a dial-up line is used and the Encode/Decode Programs are rewritten in assembly language and optimized. A complete economic evaluation requires that the cost of developing the Encoding/Decoding Tables be considered.

BIBLIOGRAPHY

1. Abramson, N., "The ALOHA System--Another Alternative for Computer Communication," AFIPS Conference Proceedings, 37:281-285, FJCC, Las Vegas, Nev., 1970.

2. _____. Information Theory and Coding. New York: McGraw-Hill, 1963.

3. "Advanced Teleprocessing Systems: Access, Allocation, Control and Security," UCLA Computer Science Quarterly , 3:45-50, July, 1975.

4. Akkoyunlu, Eralp, Arthur Bernstein, and Richard Schantz, "Interprocess Communication Facilities for Network Operating Systems," Computer, 7:46-55, June, 1974.

5. Alsberg, P. A. "Automated Resource Sharing on the ARPANET," Center for Advanced Computation (U. of Illinois) Working Paper Presented at BBN Workshop on Automated Resource Sharing, May 21, 1973.

6. Ascher, R. N. and George Nagy, "A Means for Achieving a High Degree of Compaction on Scan-Digitized Printed Text," IEEE Transactons on Computers, C-23: 1174-1179, November, 1974.

7. Ash, R. B. Information Theory, New York: Wiley, 1965.

8. Bahl, L. R. and H. Kobayashi, "Image Data Compression by Predictive Coding II: Encoding Algorithms," IBM Journal of Research and Development, 18:172-179, March, 1974.

9. Baran, P. "On Distributed Communication Networks," IEEE Transactions on Communication Systems, CS-12, March, 1964.

10. Becker, Hal B. Functional Analysis of Information Networks. New York: John Wiley & Sons, 1973.

11. Bell, C. Gordon, "More Power by Networking," IEEE Spectrum, 11:40-45, February, 1974.

12. Berger, T., F. Jelinek, and J. K. Wolf, "Permutation Codes for Sources," IEEE Transactions on Information Theory, 18:160-169, January, 1972.

13. Bobrow, Daniel G., et al., "TENEX, a Paged Time Sharing System for the PDP-10," Communications of the ACM, 15:135-143, March, 1972.

14. Booth, Grayce M., "The Use of Distributed Data Bases in Information Networks," Proceedings of the First International Conference on Computer Communication, 1972, pp. 371-376.

15. Bouknight, W. J., G. R. Grossman, D. M. Grothe, "The ARPA Network Terminal System: A New Approach to Network Access," Proceedings of the Third Data Communications Symposium, St. Petersburg, Florida, November 13-15, 1973, pp. 73-79.

16. Carlson, A. Bruce. Communication Systems: An Introduction to Signals and Noise in Electrical Communication. New York: McGraw-Hill, 1968.

17. Carr, S., S. Crocker, and V. Cerf, "Host to Host Communication Protocol in the ARPA Network," AFIPS Conference Proceedings, 36:589-597, SJCC, Atlantic City, N. J., 1970.

18. Cerf, Vinton G., Eric F. Harslem, John F. Heafner, Robert M. Metcalfe, and James E. White, "An Experimental Service for Adaptable Data Reconfiguration," IEEE Transactions on Communications, COM-20:557-563, June, 1972.

19. Chomsky, N., "Three Models for the Description of Language," IRE Transactions on Information Theory, IT-2, September, 1956.

20. Chu, W. W., "Optimal File Allocation in a Multiple Computer System," IEEE Transactions on Computers, C-18:885-889, October, 1969.

21. Chyzik, John F. "Fundamentals of Software for Communications Processors," Data Communications Systems, 2:47-56, September, 1973.

22. Cole, G. D. Computer Network Measurements: Techniques and Experiments, Engineering Report No. UCLA-ENG-7165, University of California, Los Angeles, California, 1971.

23. "Computer Network Research," UCLA Computer Science Quarterly, 2:20-23, July, 1974.

24. Crowther, W., J. McQuillan, and D. Walden, "Reliability Issues in the ARPA Network," Proceedings of the Third Data Communication Symposium, St. Petersburg, Florida, November 13-15, 1973, pp. 157-158.

25. Davies, D. W., K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson, "A Digital Communication Network for Computers Giving Rapid Response at Remote Terminals," ACM Symposium on Operating System Principles, Gatlinburg, Tenn., October, 1967.

26. Doll, Dixon R. "Computer Networking--The Giant Step in Data Communications," Data Communication Systems, 2:25-31, September, 1973.

27. _____, "How to Evaluate Service Offerings of Varied Carriers," The Data Communications User, November, 1974, pp. 16-37.

28. _____, "Multiplexing and Concentration," Proceedings of IEEE, 60:1313-1321, November, 1972.

29. _____, "Telecommunication Turbulence and the Computer Network Evolution, Computer , 11:13-22, February, 1974.

30. Dorgelo, A. J. G. and H. Van der Veer, "Variable Length Coding for Increasing Traffic Capacity in PCM Transmission Systems," IEEE Transactions on Communications, COM-21:1422-1430, December, 1973.

31. Falk, Howard, "Data Communications," IEEE Spectrum, 11:36-39, January, 1974.

32. Fano, R. M., "The MAC System: The Computer Utility Approach," IEEE Spectrum, 2:56-64, January, 1965.

33. Farber, David J., "Networks: An Introduction," Datamation , April, 1972, pp. 36-39.

34. _____, "Software Considerations in Distributed Architectures," Computer, 7:31-35, March, 1974.

35. _____, et al., "The Distributed Computer System," COMPCON '73 Digest, pp. 31-34.

36. _____, and Frank R. Heinrich, "The Structure of a Distributed Computer System--The Distributed File System," Proceedings of the International Conference on Computer Communications, 1972, pp. 364-370.

37. Felt, Douglas Charles, "Data Compression Techniques Using Adaptive Translation-Permutation Codes," PhD Dissertation, Washington State University, 1974.

38. Fisher, C. R., "Introduction to the DATRAN Switched Digital Network," Proceedings of the International Conference on Communications, June, 1971, pp. 23-1 to 23-3.

39. Frank Howard, Robert E. Kahn, and Leonard Kleinrock, "Computer Communication Network Design--Experience with Theory and Practice," AFIPS Conference Proceedings, 1972, SJCC, pp. 255-270.

40. Fultz, Gary Lee, Adaptive Routing Techniques for Message Switching Computer-Communication Networks, Engineering Report No. ULCA-ENG-7252, University of California, Los Angeles, California, 1973.

41. Gaines, Eugene C., "Specialized Common Carriers -- Competition and Alternative," Telecommunications, 7:15-26, September, 1973.

42. Gerla, Mario, The Design of Store-and-Forward (S/F) Networks for Computer Communications, Engineering Report No. UCLA-ENG-7319, University of California, Los Angeles, California, 1973.

43. Golomb, S. W., "Run-Length Encoding," IEEE Transactions on Information Theory, IT-12:399-401, July, 1966.

44. Gruenberger, F. (Ed.), Computers and Communication -- Toward a Computer Utility, Prentice-Hall, Englewood N. J., 1968.

45. Heart, F. E., R. E. Kahn, S. M. Ornestein, W. R. Crother, and D. C. Walden, "The Interface Message Processor for the ARPA Computer Network," AFIPS Conference Proceedings, 36:551-567, SJCC, Atlantic City, N. J., 1970.

46. Hirsch, Phil, "Data Communications," IEEE Spectrum, 8:47-60, February, 1971.

47. _____, "SITA: Rating a Packet Switched Network," Datamation, 20:60-63, March, 1974.

48. Hopcroft, John E. and Jeffery D. Ullman, Formal Languages and Their Relation to Automata . Reading, Mass.: Addison-Wesley Publishing Co., 1969.

49. Hu, T. C. and A. C. Tucker, "Optimal Computer Search Trees and Variable Length Alphabetic Codes," SIAM Journal of Applied Math, 21:514-532, October, 1971.

50. Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes," Proceedings of IRE, 40:1098-1101, September, 1952.

51. Ives, Fredrick M., "Interactive File Access in a Batch-Oriented Computer System," PhD Dissertation, Washington State University, 1972.

52. James, R. T. and Muench, P. E., "AT&T Facilities and Services," Proceedings of IEEE, 60:1342-1349, November, 1972.

53. Kahn, Robert E., "Resource-Sharing Computer Communications Networks," Proceedings of IEEE, 60:1397-1407, November, 1972.

54. _____, and William R. Crowther, "Flow Control in a Resource Sharing Computer Network," IEEE Transactions on Communications, COM-20:539-546, June, 1972.

55. Kirlin, R. L., "Variable Block Length and Transmission Efficiency," IEEE Transactions on Communications Technology, COM-17:340-349, June, 1969.

56. Knight, John R., "A Case Study: Airlines Reservation Systems," Proceedings of the IEEE, 60:1423-1431, November, 1972.

57. Kobayashi, H. and L. R. Bahl, "Image Data Compression by Predictive Coding 1: Prediction Algorthms," IBM Journal of Research and Development, 18:164-171, March, 1974.

58. Lynch, Thomas J., "Comparison of Time Codes for Source Encoding," IEEE Transactions on Communications, 22:151-162, February, 1974.

59. Martin, James. Systems Analysis for Data Transmission.

Englewood Cliffs, N. J.: Prentice Hall, Inc., 1972.

60. _____, Telecommunications and the Computer. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1969.

61. Mastromonaco, F. R., "Optimum Speed of Service in the Design of Customer Data Communication Systems," ACM Symposium on Problems in the Optimization of Data Communication Systems, Pine Mountain, Georgia, October 13-16, 1969, pp. 129-153.

62. Mathison, S. L. and P. M. Walker, "Regulatory and Economic Issue in Computer Communications," Proceedings of IEEE, 11:1254-1272, November, 1972.

63. McKee, David J. "How Transaction Cost Decline as Data Networks Get Larger," Data Communication Systems, 2:17-22, September, 1973.

64. _____, and Terence J. McCormick, "Wanted: A Computer/Communications Resale Industry," Data Communications, 3:30-36, July/August, 1974.

65. Mehta, Kirit and Carl N. Boustead, "Getting Peak Performance on a Data Channel," Data Communications, 3:39-47, July/August, 1974.

66. Meister, B., H. R. Mueller, and H. R. Rudin, Jr., "On the Optimization of Message-Switching Networks," IEEE Transactions on Communications, COM-20:8-14, February, 1972.

67. Meyr, H., H. G. Rosdolsky, and T. S. Huang, "Optimum Run-Length Codes," IEEE Transactions on Communications, COM-22:826-835, June, 1974.

68. Mills, David L. "Communication Software," Proceedings of IEEE, 60:1333-1341, November, 1972.

69. Oliver, B. M., "Efficient Coding," Bell System Technical Journal, 31:724-750, July, 1952.

70. Ornstein, S. M., F. E. Heart, W. R. Crowther, H. K. Rising, S. B. Russell, and A. Michael, "The Terminal IMP for the ARPA Computer Network, AFIPS Conference Proceedings, 40:243-254, SJCC, 1972.

71. Ott, Gene, "Compact Encoding of Stationary Markov Sources," IEEE Transactions on Information Theory , IT-13: 82-86, January, 1967.

72. "Packet Switching," Telecommunications Handbook & Buyer's Guide, September, 1974, pp., 26-27.

73. Pyke, T. N., Jr. and R. P. Blanc, "Computer Networking Technology -- A State of the Art Review," Computer, 6:13-19, August, 1973.

74. "Rating Carriers on Basis of Cost abd Extent of Services," The Data Communication User, November, 1975, pp. 28-43.

75. Reza, Fazlollah M., Introduction to Information Theory. New York: McGraw-Hill, 1961.

76. Roberts, Lawrence G. "Data by the Packet," IEEE Spectrum, 11:46-51, February, 1974.

77. _____, "Network Rational: A 5-Year Reevaluation", Proceedings of COMPCON , 1973, pp. 3-5.

78. _____, and B. D. Wessler, "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, 36:543-549, SJCC, Atlantic City, N. J., 1970.

79. Shaft, Paul D., "A Source Encoding Algorithm for Quantized Data," IEEE Transactions on Communications, COM-22:867-869, June, 1974.

80. Shannon, C. E., "Prediction and Entropy of Printed English," Bell System Technical Journal, 30:50-64, January, 1951.

81. _____, and W. Weaver. The Mathematical Theory of Communication. Urbana, Illinois: University of Illinois Press, 1949.

82. Sher, Michael S. "A case Study in Networking," Datamation, 20:56-59, March, 1974.

83. Smith, Stephen A., "A Generalization of Huffman Coding for Messages with Relative Frequencies Given by Upper and Lower Bounds," IEEE Transactions on Information Theory, IT-20:124-125, January, 1974.

84. "Specialized Communications," Telecommunications, 8:19-34, September, 1974.

85. Tat, Than N'Guyen and Anatole Girinsky, "Evolution of Store-and-Forward Techniques," IEEE Transactions on

Communications, 22:1297-1301, September, 1974.

86. Thomas, Robert H., "A Resource Sharing Executive for the ARPANET," Proceedings of National Computer Conference, 42:155-163, 1973.

87. Thompson, Richard Arthur, Compact Encoding of Probabilistic Languages, Underwater Sound Laboratory Research Project Report USL-5, University of Connecticut School of Engineering, Storrs, Connecticut.

88. Titus, James P. "Data Communications -- Issues and More Issues," Communication of ACM, 17:120-122, February, 1974.

89. Turner, I. F., "Data Compression Techniques as A Means of Reducing the Storage Requirements for Satellite Data: A Quantitative Comparison," Radio Electronic Engineering (GB), Vol. 43, October, 1973.

90. Voorhis, David C., "Constructing Codes with Bounded Codeword Lengths," IEEE Transactions on Information Theory, IT-20:288-291, March, 1974.

91. Walden, D. "A System for Interprocess Communication in a Resource Sharing Computer Network," Communication of the ACM, 15:221-230, April, 1972.

92. Weingarten, A., "Storage Requirements for a Message Switching Computer," IEEE Transactions on Communication Systems, CS-12:191-195, June, 1964.

93. Whitney, V. Kelvin Moore, A Study of Optimal File Assignment and Communication Network Configuration in Remote Access Computer Message Processing and communication Systems, PhD Dissertation, University of Michigan, SEL Technical Report No. 48, September, 1970.

94. Withington, Frederic G., "Five Generations of Computers," Harvard Business Review, 52:99-108, July-August, 1974.

95. Worley, A. R., "The DATRAN System," Proceedings of the IEEE, 60:1357-1368, November, 1972.

APPENDIX A. SERIES 6000 SIX–BIT BCD CHARACTER SET

Standard Character Set	Internal Machine Code	Octal Code	Hollerith Card Code	Standard Character Set	Internal Machine Code	Octal Code	Hollerith Card Code
0	00 0000	00	0	↑	10 0000	40	11-0
1	00 0001	01	1	J	10 0001	41	11-1
2	00 0010	02	2	K	10 0010	42	11-2
3	00 0011	03	3	L	10 0011	43	11-3
4	00 0100	04	4	M	10 0100	44	11-4
5	00 0101	05	5	N	10 0101	45	11-5
6	00 0110	06	6	O	10 0110	46	11-6
7	00 0111	07	7	P	10 0111	47	11-7
8	00 1000	10	8	Q	10 1000	50	11-8
9	00 1001	11	9	R	10 1001	51	11-9
[	00 1010	12	2-8	←	10 1010	52	11
#	00 1011	13	3-8	$	10 1011	53	11-3-8
@	00 1100	14	4-8	*	10 1100	54	11-4-8
:	00 1101	15	5-8	)	10 1101	55	11-5-8
>	00 1110	16	6-8	;	10 1110	56	11-6-8
?	00 1111	17	7-8	'	10 1111	57	11-7-8
b	01 0000	20	(blank)	+	11 0000	60	12-0
A	01 0001	21	12-1	/	11 0001	61	0-1
B	01 0010	22	12-2	S	11 0010	62	0-2
C	01 0011	23	12-3	T	11 0011	63	0-3
D	01 0100	24	12-4	U	11 0100	64	0-4
E	01 0101	25	12-5	V	11 0101	65	0-5
F	01 0110	26	12-6	W	11 0110	66	0-6
G	01 0111	27	12-7	X	11 0111	67	0-7
H	01 1000	30	12-8	Y	11 1000	70	0-8
I	01 1001	31	12-9	Z	11 1001	71	0-9
&	01 1010	32	12	←	11 1010	72	0-2-8
.	01 1011	33	12-3-8	,	11 1011	73	0-3-8
]	01 1100	34	12-4-8	%	11 1100	74	0-4-8
(	10 1101	35	12-5-8	=	11 1101	75	0-5-8
<	01 1110	36	12-6-8	"	11 1110	76	0-6-8
!	01 1111	37	12-7-8	!	11 1111	77	0-7-8

APPENDIX B. SOURCE PROGRAMS

NRAP ABSTRACT

1.0 IDENTIFICATION DESCRIPTION

1.1 LIBRARY- NONE
1.2 TITLE- JCL REDUNDANCY PROGRAM (NRAP)
1.3 SOURCE- FORTRAN(MAIN) & GMAP(SUBROUTINE)
1.4 INTERFACE- NONE
1.5 AUTHOR- W. R. BODWELL (1-804-380-7670)

2.0 PURPOSE

THE NRAP PROGRAM CREATES TABLES FOR CALCULATING HC0 AND HC1, AND GENERATES APPROPRIATE ENCODING/DECODING TABLES. THE ENCODING/DECODING TABLES ARE UTILIZED TO COMPRESS THE DATA CONTAINED WITHIN THE FILES BY REMOVING REDUNDANCY BY PROPER ASSIGNMENT OF SYMBOLS TO REPRESENT STRINGS AND SYMBOLS. INTERMEDIARY TABLES USED BY THE RAP PROGRAM ARE DESCRIBED BELOW:

INITIAL/REPEAT DATA TABLE (IRD) IRD = 64 X 80 = 5,120 WDS

IRD(I,1) = NUMBER OF TIMES SYMBOL APPEARS AS FIRST CHAR.
IRD(I,2) = NUMBER OF TIMES SYMBOL I-1 REPEATED 2 TIMES
IRD(I,3) = NUMBER OF TIMES SYMBOL I-1 REPEATED 3 TIMES
. . .
IRD(I,80) = NUMBER OF TIMES SYMBOL I-1 REPEATED 80 TIMES

WHERE 0 (I (65

FINAL/FREQUENCY DATA TABLE (FFD) FFD = 65 X 66 = 4,290 WDS.

FFD(I,1) = NUMBER OF TIMES SYMBOL APPEARED AS 80TH CHAR.
FFD(I,2) = NUMBER OF TIMES SYMBOL I FOLLOWED BY SYMBOL 0
- - -
FFD(I,65) = NUMBER OF TIMES SYMBOL I FOLLOWED BY SYMBOL ½
FFD(65,J) = SUM OF NUMBERS WITHIN RESPECTIVE COLUMNS
FFD(I,66) = SUM OF NUMBERS WITHIN RESPECTIVE ROWS

WHERE 0 (I (65
0 (J (66

FFD(65,66) = TOTAL NUMBER OF SYMBOLS PROCESSED
= SUM OF FFD(I,66) 0 (I (65
= SUM OF FFD(65,J) 0 (J (66

ICOUNT TABLE (IT) CONTAINS CANDIDATES FOR THE ENCODING/ DECODING TABLE (EDT). ICOUNT = 200 X 8 = 1,600 WDS.

ICOUNT(I,1) = ENTRY TYPE
- 1 = STRING SYNTAX FROM SLST
- 2 = STRINGS FROM SLST
  STRINGS GENERATED USING 1
- 3 = REPEATED CHARACTER STRINGS
- 4 = SELECTED STRINGS FROM 2
- 5 = ORIGINAL SYMBOLS
- 6 = END-OF-RECORD CHARACTER
- 7 = REPEAT CHARACTER
- 8 = TABLE SHIFT CHARACTER(S)

ICOUNT(I,2) = POSITION OR WEIGHT
ICOUNT(I,3) = NUMBER OF CHARACTER IN STRING
ICOUNT(I,4) = STRING SYMBOLS, 1-6
ICOUNT(I,5) = STRING SYMBOLS, 7-12
ICOUNT(I,6) = STRING SYMBOLS, 13-18
ICOUNT(I,7) = STRING SYMBOLS, 19-24
ICOUNT(I,8) = NUMBER OF TIMES STRING OR SYMBOL ENCOUNTERED

IRANK IS A BUFFER USED TO SORT ICOUNT. IRANK = 200 X 8

SORTED BY CALCULATED WEIGHT

H(C0) = ENTROPY (BITS/SYMBOL) ; SYMBOLS INDEPENDENT

H(C1) = CONDITIONAL ENTROPY (BITS/SYMBOL) ; SYMBOL J DEPENDENT UPON 1 PRECEEDING SYMBOL I

3.0 FUNCTION

THE NRAP PROGRAM SEARCHES ALL JCL FILES(FC-20) ENUMERATED WITHIN A FILENAME FILE (FC-10) TO CALCULATE ENTROPY, H(C0), AND CONDITIONAL ENTROPY, H(C1), AND TO CREATE AN ENCODING/DECODING TABLE WHICH IS OUTPUT TO FC-30.

4.0 BACKGROUND FUNCTION

4.1 HISTORY OF DEVELOPMENT

07/74 TO 09/74 — ZJCLE PROGRAM DEVELOPED TO CREATE DATABASE CONTAINING JCL FILENAME AND TAPE FILENAME. THE DATABASE ENTRIES ARE USED TO CREATE A JCL FILENAME/TAPE FILENAME CROSS-REFERENCE LIST

09/74 — MAJOR REVISION OF ZJCLE PROGRAM. PURPOSE

TO 12/74 CHANGED TO CALCULATE HC0 AND HC1. ZJCLE'S TECHNIQUE'S FOR ACCESSING, SEARCHING, AND DEACCESSING FILES REMAIN UNCHANGED. PROGRAM ZJCLE =) ZJCLR

01/75 TO 01/76 SECOND MAJOR REVISION OF PROGRAM TO SUPPRESS TRAILING BLANKS. PROGRAM FURTHER MODIFIED TO UTILIZED THE INFORMATION CONTAINED IN FFD, ICOUNT, AND IRD TO AUTOMATICALLY GENERATE APPROPRIATE ENCODING/ DECODING TABLES. PROGRAM ZJCLR =) NRAP

4.2 COMPANY & DEPARTMENT SUPPLYING- NNSDDC 594B & C66
4.3 ITEM COMPANY & DEPARTMENT USING- NNSDDC 594B & C66
4.4 RELATED DOCUMENTS- W. R. BODWELL'S DESK

5.0 RESTRICTIONS DESCRIPTION

5.1 EXTERNAL DECLARATIONS
- FC-06 OUTPUT LIST FILE
- FC-10 INPUT FILENAME FILE
- FC-30 SOURCE LANGUAGE SYNTAX TABLE
- FC-40 OUTPUT SLST AND ENCODING/DECODING TABLES

5.2 MEMORY REQUIREMENTS-26K
5.3 ADDITIONAL RESTRICTIONS- NONE

6.0 OPERATING PROCEDURE

6.1 ITEM PRELIMINARY PREPARATION

PROGRAM ZCATA OR TAFT'S ACTIVITY MUST CREATE AN INPUT FILENAME FILE.

6.2 LOADING PROCEDURE-

BATCH
=====

```
$       PROGRAM  NRAP
$       PRMFL    **,R,R,POBJUMC/ONRAP
$       PRMFL    H*,R,R,POBJUMC/ONRAP
```

TIME SHARE
==========

SYSTEM ? ZZZSBODFILES/JCLR/ONRAP

6.3 INITIALIZATION- NONE
6.4 EXIT AND TERMINATION CONDITIONS- END-OF-FILE ON FC-10, ENDS FIRST PHASE OF PROGRAM STATISTICS GATHERING.

7.0 OPERATING CHARACTERISTICS

8.0 ERROR CONDITIONS AND RECOVERIES

AT COMPLETION OF THE NRAP PROGRAM, ALL FILES WHICH WERE NOT PROCESSED CORRECTLY WILL BE LISTED ALONG WITH THE ERROR CONDITIONS AS DEFINED IN THE TIME SHARING SYSTEM PROGRAMMERS' REFERENCE MAMUAL, DD-17, PAGE 3-38. APPROPRIATE ACTION SHOULD BE TAKEN AFTER ANALYSIS OF THE ERROR.

PROGRAM DESCRIPTION

A FILENAME IS READ FROM THE INPUT FILENAME FILE(FC-10). THE FILE IS ATTACHED TO THE PROGRAM AS FC-20. A GMAP SUBROUTINE IS USED TO REFORMAT ALL CARDS IMAGES. THE SYMBOLS, I.E., CHARACTERS, ARE PROCESSED TO CREATE AN INITIAL/REPEAT DATA TABLE (IRD), AND ALSO CREATES A FINAL/FREQUENCY DATA TABLE (FFD). STRINGS SATISFYING THE STRING SYNTAX IN SLST AND THE ENUMERATED STRINGS ARE ALSO COUNTED IN TABLE ICOUNT. THE INITIAL/REPEAT DATA TABLE, FINAL/FREQUENCY DATA TABLE, AND ICOUNT TABLE ARE UTILIZED TO GENERATE AN APPROPRIATE ENCODING/DECODING TABLE. WHEN AN END-OF-FILE INDICATOR IS ENCOUNTERED ON THE FILENAME FILE (FC-10), THE INDIVIDUAL SYMBOL SUM ARE TOTALED. MAXIMUM ENTROPY, HMAX; ENTROPY, H(C0); AND CONDITIONAL ENTROPY, H(C1) ARE CALCULATED.

REDUNDANCY ANALYSIS PROGRAM

```
1*#RUNH /JCLR/SRAP;/JCLR/FILFR=/JCLR/ONRAP(BCD,CORE=20); #06
2C
3         DIMENSION IJCLFN(9),ITAPFN(9),JCLFN(2),JTAPFN(2)
4         DIMENSION INPUT(80),IREPT(30),IRD(64,80),FFD(65,66)
5         DIMENSION IDATE(2),ICOUNT(200,8),ISTRNG(3),ICST(10,24)
6         DIMENSION IRANK(200,8),ITBS(5),ITAFLG(4)
7         CHARACTER CSTRNG*24
8         EQUIVALENCE (CSTRNG,ISTRNG)
9         EQUIVALENCE (IJCLFN(5),JCLFN(1)),(ITAPFN(5),JTAPFN(1))
10         INTEGER FFD
11         DATA IBLANK/6H      /
12         DATA IGOST/O400000000000/
13         DATA IMOD/1/
14         DATA IPRMIS/1/
15         DATA IRPCT/1/
16         DATA ISTAT/0/
17         DATA (ITAPFN(I),I=1,4)/6HTCNJCL,2*6H      ,6H    / /
18         DATA (ITAPFN(I),I=7,9)/6H      ,6H      ,6H"20" ;/
19         DATA ITIF1/6HTABLE /
20         DATA ITIF2/6HSHIFT /
21         DATA IZERBK/6H0     /
22         DATA LEOR/6HEOR   /
23         DATA LREP/6HREP   /
24         DATA I20/20/
25         DATA I37/37/
26     1   FORMAT(2A6,I6)
27     2   FORMAT(4(16I8/)/)
28     3   FORMAT(4(16I8/),I8//)
29     5   FORMAT(14A6)
30     6   FORMAT(1X,14A6)
31     7   FORMAT(13HCONTROL CARDS,7X,I8/18HCOMMENT/DATA CARDS,/
32      &    2X,I8,13HTOTAL CARDS  ,7X,I8///)
33     8   FORMAT(1X,2A6,O12)
34     9    FORMAT(1X,A6,A2,2X,F6.3)
35    10   FORMAT(15(4(1X,R1,3H = ,I8,3X)/),3(1X,R1,3H = ,I8,3X),
36      &   5H ½ = ,I8//1X,6HSUM = ,I7,2X,12HEOR CHAR. = ,I6,
37      &   2X,15HBLANKS SUPP. = ,I7//)
38    11   FORMAT(7HHMAX = ,F8.4,4X,6HHC0 = ,F8.4,4X,6HHC1 = ,
39      &     F8.4///)
40    12   FORMAT(4(1X,R1,3H = ,I8,3X))
41    13   FORMAT(1X,I8)
42    14   FORMAT(3I4,A24)
43    15   FORMAT(24R1)
44    16   FORMAT(21H  TP   WGH CH  STRING,19X,9HFREQUENCY)
45    17   FORMAT(1X,I3,I6,I3,2X,4A6,2X,I8)
46    18   FORMAT(6HHC0 = ,F8.4,2X,"BLANKS REMOVED"//)
47    19   FORMAT(6HHC0 = ,F8.4,2X,"BLANKS REMOVED - STRINGS",
48      &   "SUBSTITUTED"//)
49    20   FORMAT(//"NUMBER OF TABLES = ",I2)
50    21   FORMAT(8(I6,2X))
```

```
51    22   FORMAT(//"INC. STRINGS SELECTED = ",I3,3X,"TOTAL "
52      &   "STRINGS SELECTED = ",I3//)
53    23   FORMAT(//"UNSELECTED REPEATED STRINGS",I6//)
54    24   FORMAT(/"# OF ENTRIES IN TABLE ",I1, " IS ",I6//)
55    25   FORMAT(/"AVERAGE OF ",F6.4," BITS/SYMBOL"/)
56    26   FORMAT(" ")
57    27   FORMAT(//4X,"TABLE 1"/)
58    28   FORMAT(//4X,"TABLE 2"/)
59    29   FORMAT(//4X,"TABLE ",I1/)
60    31   FORMAT(//"INPUT FILE STATISTICS"/)
61    32   FORMAT(//"INTERMEDIATE STATISTICS"/)
62    33   FORMAT(//"ENCODING/DECODING TABLES"/)
63    34   FORMAT(//"OUTPUT STATISTICS - BLANKS REMOVED & ",
64      &  "STRINGS SUBSTITUTED")
65    36   FORMAT(//"SYMBOLS AND STRINGS BEFORE ASSIGNMENT"/)
66    37   FORMAT(//"SOURCE LANGUAGE SYNTAX TABLE"/)
67    38   FORMAT(4X,R1,3X,I3,I6,I3,2X,4A6,2X,I8)
68C
69C        HEAD
70C
71C        CALL LLINK("*NRAP")
72         PRINT ,"PROGRAM START"
73         CALL DATIM(IDATE,TIME)
74         PRINT 9,IDATE,TIME
75         PRINT ," "
76C
77C        READ SOURCE LANGUAGE SYNTAX TABLE
78C
79C        FC 36 =) SOURCE LANGUAGE SYNTAX TABLE
80C
81         WRITE(6,37)
82         WRITE(6,17)
83         ICTLOC = 2
84    30   READ(30,14,END=45)(ICOUNT(ICTLOC,IX),IX=1,3),CSTRNG
85         IF((ICOUNT(ICTLOC,2)+ICOUNT(ICTLOC,3)).LE.81)
86       &    GO TO 35
87         PRINT,"SYNTAX CARD ERROR -- DELETED"
88         GO TO 30
89    35   DECODE(CSTRNG,15) (ICST(ICTLOC,IX),IX=1,24)
90         DO 40 I=1,4
91    40   ICOUNT(ICTLOC,3+I) = ISTRNG(I)
92         WRITE(6,17) (ICOUNT(ICTLOC,IY),IY=1,8)
93         WRITE(40,17) (ICOUNT(ICTLOC,IY),IY=1,8)
94         ICTLOC = ICTLOC + 1
95         GO TO 30
96    45   ICTLST = ICTLOC - 1
97         ICTNXT = ICTLOC
98C
99C        JCLR PROGRAM
100C
```

```
101C        FC 06  =)  SOURCE LANGUAGE SYNTAX TABLE
102C                   INPUT FILE STATISTICS
103C                   INTERMEDIATE STATISTICS
104C                   ENCODING/DECODING TABLES
105C                   OUTPUT STATISTICS - BLANKS REMOVED
106C                      AND STRINGS SUBSTITUTED
107C        FC 10  =)  INPUT FILENAME FILE
108C        FC 20  =)  USED TO ATTACH,READ,DETACH JCL FILES
109C        FC 30  =)  SOURCE LANGUAGE SYNTAX TABLE
110C        FC 40  =)  SLST AND ENCODING/DECODING TABLES
111C        FC 37 =) FFD DATA
112C
113    50   READ(10,1,END=115) JTAPFN,LLSIZ
114         CALL ATTACH(I20,ITAPFN,IPRMIS,IMOD,ISTAT,)
115         IF(ISTAT.EQ.0) GO TO 58
116         IF(ISTAT.EQ.IGOST) GO TO 58
117C
118C         ERROR ON FC 20
119C
120    55   IREPT(IRPCT) = ITAPFN(5)
121         IREPT(IRPCT+1) = ITAPFN(6)
122         IREPT(IRPCT+2) = ISTAT
123         IRPCT = IRPCT + 3
124         IF(IRPCT.GT.29) IRPCT = 28
125         GO TO 50
126C
127C         GMAP PROGRAM READS AND REFORMATS CARD IMAGE
128C
129    58   ICARCT = 0
130    60   CALL REFORM(INPUT,LLSIZ)
131C
132C          LLSIZ: (0 =) READ ERROR; =0 =) SEARCH COMPLETE
133C         LLSIZ)0  =) CHECK SECONDARY CONDITIONS
134C
135         IF(LLSIZ.GT.0) ICARCT = ICARCT + 1
136         IF(LLSIZ) 62,62,64
137C
138C         DETACH FILE
139C
140   62   CALL DETACH(I20,ISTAT,)
141         IF(ISTAT.EQ.IGOST) GO TO 50
142         IF(ISTAT.EQ.0) GO TO 50
143         GO TO 55
144C
145C         SUPPRESS TRAILING BLANKS
146C
147    64   DO 66 ICT=1,80
148         IF(INPUT(81-ICT).NE.16) GO TO 68
149    66   CONTINUE
150    68   IF(ICT.EQ.1) GO TO 69
```

```
151          ILAST = 82 - ICT
152          IEORFL = 1
153          IEOR = IEOR + 1
154          IBLKSV = IBLKSV + 80 - ILAST
155          GO TO 70
156     69   ILAST = 80
157          IEORFL = 0
158C
159C          CHECK TO SEE IF VARIABLE OR FIXED
160C          SYNTAX PATTERN ON THIS CARD IMAGE
161C
162     70   IMIN = 1
163          DO 71 IPOS=1,5
164     71   ITBS(IPOS) = 0
165          IPOS = 1
166          DO 90 ICKLOY = 2,ICTLST
167          IF(ICOUNT(ICKLOY,2).EQ.0) IMGLOX = 0
168          IF(ICOUNT(ICKLOY,2).NE.0) IMGLOY =
169       &     ICOUNT(ICKLOY,2) - 1
170     72   DO 75  ICKLOX = 1,ICOUNT(ICKLOY,3)
171          IF(ICST(ICKLOY,ICKLOX).EQ.11) GO TO 75
172          IF(ICST(ICKLOY,ICKLOX).NE.INPUT(IMGLOX+ICKLOX))
173       &        GO TO 78
174     75   CONTINUE
175          GO TO 80
176     78   IF(ICOUNT(ICKLOY,2).EQ.0) IMGLOX = IMGLOX+1
177          IF((ICOUNT(ICKLOY,2).EQ.0).AND.(IMGLOX.LE.(81-
178       &     ICOUNT(ICKLOY,3)))) GO TO 72
179          GO TO 90
180C
181C          MATCH (INCREMENT PROPER COUNTER)
182C
183     80   IMAX = IMGLOX+ICOUNT(ICKLOY,3)
184          IF(IMAX.GT.IMIN) IMIN = IMAX
185          IF(INPUT(IMAX).EQ.16) ITBS(IPOS) = IMAX + 1
186          IF((INPUT(IMAX).EQ.16).AND.(IPOS.LE.4))IPOS=IPOS+1
187          IF(ICARCT.EQ.1) GO TO 81
188          IF(ICOUNT(ICKLOY,2).NE.1) GO TO 81
189          IF(IEORFL.NE.1) GO TO 81
190           IEORS = IEORS + 1
191     81   GO TO (82,87), ICOUNT(ICKLOY,1)
192     82   ENCODE(CSTRNG,15) (INPUT(IMGLOX+ICT),ICT=1,ICOUNT
193       &   (ICKLOY,3)),(IBLANK,I=1,24-ICOUNT(ICKLOY,3))
194          IF(ICOUNT(ICTLST+1,1).EQ.0) GO TO 88
195          DO 84 ICTLOC=ICTLST+1,ICTNXT-1
196          IF(ICOUNT(ICTLOC,4).NE.ISTRNG(1)) GO TO 84
197          IF(ICOUNT(ICTLOC,5).NE.ISTRNG(2)) GO TO 84
198          IF(ICOUNT(ICTLOC,6).NE.ISTRNG(3)) GO TO 84
199          IF(ICOUNT(ICTLOC,7).EQ.ISTRNG(4)) GO TO 86
200     84   CONTINUE
```

```
201         GO TO 88
202C
203C        MATCH FOUND
204C
205    86   ICOUNT(ICTLOC,8) = ICOUNT(ICTLOC,8) + 1
206         GO TO 90
207    87   ICOUNT(ICKLOY,8) = ICOUNT(ICKLOY,8) + 1
208         GO TO 90
209C
210C        CREATE NEW VARIABLE PATTERN
211C
212    88   ICOUNT(ICTNXT,1) = 2
213         ICOUNT(ICTNXT,2) = ICOUNT(ICKLOY,2)
214         ICOUNT(ICTNXT,3) = ICOUNT(ICKLOY,3)
215         ICOUNT(ICTNXT,4) = ISTRNG(1)
216         ICOUNT(ICTNXT,5) = ISTRNG(2)
217         ICOUNT(ICTNXT,6) = ISTRNG(3)
218         ICOUNT(ICTNXT,7) = ISTRNG(4)
219         ICOUNT(ICTNXT,8) = 1
220         ICOUNT(ICKLOY,8) = ICOUNT(ICKLOY,8) + 1
221         ICTNXT = ICTNXT + 1
222         IF(ICTNXT.EQ.193) PRINT, "TOO MANY STRINGS FOUND"
223         IF(ICTNXT.EQ.193) GO TO 990
224    90   CONTINUE
225C
226C        CARD IMAGE PROCESSING (INPUT(I),I=1,ILAST)
227C        CREATION OF IRD(64,80) AND FFD(65,66) TABLES
228C
229         IF(ILAST.GE.IMIN) GO TO 95
230         IBLKSV = IBLKSV - (IMIN-ILAST)
231         ILAST = IMIN
232    95   INPUT(ILAST) = 63
233         DO 105  I=1,ILAST
234         IF(I.NE.1) GO TO 100
235         IRD(INPUT(I)+1,1) = IRD(INPUT(I)+1,1) + 1
236         IREP = 0
237         GO TO 105
238   100   IF(INPUT(I-1).EQ.INPUT(I)) GO TO 105
239         IF(IREP.GT.1) IRD(INPUT(I-1)+1,IREP) =
240      &        IRD(INPUT(I-1)+1,IREP) + 1
241         IREP = 0
242   105   IREP = IREP + 1
243         INDEX = INPUT(ILAST) + 1
244         IF(IREP.GT.1) IRD(INDEX,IREP) = IRD(INDEX,IREP)+1
245         DO 110 I=2,ILAST
246         FFD(INPUT(I-1)+1,INPUT(I)+2) = FFD(INPUT(I-1)+1,
247      &       INPUT(I)+2) + 1
248   110   CONTINUE
249         ILASPR = INPUT(ILAST) + 1
250         FFD(ILASPR,1) = FFD(ILASPR,1) + 1
```

```
251         GO TO 60
252C
253C        CALCULATE CARD TYPE
254C
255   115   DO 120  I=1,64
256   120   IREC = IRD(I,1) + IREC
257         ICOMDA = IREC - IRD(44,1)
258C
259C        PRINT CARD TYPE
260C
261         WRITE(6,31)
262         WRITE(6,17)
263         WRITE(6,7) IRD(44,1),ICOMDA,IREC
264C
265C         CALCULATE SUM OF SYMBOL OCCURRENCES
266C
267         DO 135   I=1,64
268         DO 130   J=1,65
269   130   FFD(I,66) = FFD(I,J) + FFD(I,66)
270   135   CONTINUE
271C
272         DO 145   J=1,66
273         DO 140   I=1,64
274   140   FFD(65,J) = FFD(I,J) + FFD(65,J)
275   145   CONTINUE
276C
277         DO 150   J=1,65
278   150   ICHECK = ICHECK + FFD(65,J)
279         IF(ICHECK.NE.FFD(65,66)) PRINT, "SUM ERROR"
280C
281C        WRITE IRD(64,80) AND FFD(65,66) TABLES
282C
283         WRITE(6,10)((I-1,FFD(I,66)),I=1,63),FFD(64,66),
284        &   FFD(65,66),IEOR,IBLKSV
285         ITOTIN = FFD(65,66)
286C
287C        CALCULATE HMAX
288C
289         HMAX = ALOG(64)/ALOG(2)
290C
291C        CALCULATE ENTROPY, HC0   (BLANKS SUPPRESSED)
292C
293         DO 160   I=1,64
294         IF(FFD(I,66).EQ.0) GO TO 160
295         HC0S = HC0S+FFD(I,66)*ALOG(FFD(65,66)/FFD(I,66))
296   160   CONTINUE
297         HC0 = HC0S/ALOG(2)/FFD(65,66)
298C
299C        CALCULATE CONDITIONAL ENTROPY, HC1
300C
```

```
301          DO 170   I=1,64
302          DO 165   J=1,65
303          IF(FFD(I,J).EQ.0) GO TO 165
304          HC1S = HC1S+FFD(I,J)*ALOG(FFD(I,66)/FFD(I,J))
305    165   CONTINUE
306    170   CONTINUE
307          HC1 = HC1S/ALOG(2.)/FFD(65,66)
308C
309C          PRINT H(MAX), ENTROPY, COND. ENTROPY
310C
311          WRITE(6,11)  HMAX,HC0,HC1
312          FFD(64,66) = FFD(64,66) - IEOR
313          FFD(65,66) = FFD(65,66) - IEOR
314          IEOR = IEOR - IEORS
315C
316C          PRINT FILENAMES NOT CORRECTLY PROCESSED
317C
318          IF(IRPCT.EQ.1) GO TO 190
319          PRINT ," "
320          PRINT ," "
321          PRINT ,"ABNORMALITIES"
322          PRINT ," "
323          DO 180 I=1,IRPCT-1,3
324    180   PRINT 8,IREPT(I),IREPT(I+1),IREPT(I+2)
325    190   CONTINUE
326C
327C          OUTPUT VARIABLE AND CONSTANT INPUT STRINGS
328C
329C         WRITE(30,17)((ICOUNT(ICT,JCT),JCT=1,8),ICT=2,ICTLST)
330C
331C          ADJUST SORT VARIABLES
332C
333          ISELEN = 0
334          ILSTNG = ICTNXT - 1
335C
336C          CALCULATE WEIGHTS FOR ICOUNT AND IRD
337C
338          DO 200 IWT=1,ICTNXT-1
339          IF(IWT.GT.ICTLST) GO TO 195
340          IF(ICOUNT(IWT,1).NE.1) GO TO 195
341          ICOUNT(IWT,1) = 0
342          ICOUNT(IWT,2) = 0
343          ICOUNT(IWT,3) = 0
344          ICOUNT(IWT,8) = 0
345          DO 193 ICT=4,7
346    193   ICOUNT(IWT,ICT) = IBLANK
347          GO TO 200
348    195   ICOUNT(IWT,2) = (ICOUNT(IWT,3)-1)*ICOUNT(IWT,8)
349    200   CONTINUE
350          GO TO 204
```

```
351  202    ICTNXT = ILSTNG + 1
352         GO TO 205
353C
354C        CALCULATE WEIGHTS FOR REPEATED CHARACTERS FOR IRD
355C
356  204    ISETBL = FFD(65,66)/64/64
357         IF(ISETBL.LT.1) ISETBL = 1
358         ISECRI = ISETBL
359  205    DO 220 JCT = 3,80
360         DO 210 ICT=1,64
361         IF(IRD(ICT,JCT).LT.ISECRI) GO TO 210
362         IF(JCT.LE.3) IWT = JCT -1
363         IF(JCT.GT.3) IWT = 2
364         IF(JCT.GE.64) IWT = 3
365         IF(JCT.LE.23) ENCODE(CSTRNG,15) (ICT-1,I=1,JCT),
366      &   (IBLANK,I=1,24-JCT)
367         IF(JCT.GT.23) ENCODE(CSTRNG,15) (ICT-1,I=1,24)
368         ICOUNT(ICTNXT,1) = 3
369         ICOUNT(ICTNXT,2) = IWT*IRD(ICT,JCT)
370         ICOUNT(ICTNXT,3) = JCT
371         ICOUNT(ICTNXT,4) = ISTRNG(1)
372         ICOUNT(ICTNXT,5) = ISTRNG(2)
373         ICOUNT(ICTNXT,6) = ISTRNG(3)
374         ICOUNT(ICTNXT,7) = ISTRNG(4)
375         ICOUNT(ICTNXT,8) = IRD(ICT,JCT)
376         ICTNXT = ICTNXT+1
377         IF(ICTNXT.EQ.201) GO TO 230
378  210    CONTINUE
379  220    CONTINUE
380  230    CONTINUE
381C
382C         CALCULATE NO. OF TABLES TO BE USED
383C
384         IF(ITAFLC.EQ.1) GO TO 234
385         ITAFLC = 1
386         DO 232 ICT=1,ICTNXT-1
387         IF(ICOUNT(ICT,2).GT.ISETBL) IPOENT = IPOENT + 1
388  232    CONTINUE
389         INOTBL = 2 + IPOENT/64
390         IF(INOTBL.GT.3) INOTBL = 3
391         WRITE(6,32)
392         WRITE(6,20) INOTBL
393C
394C         ENOUGHT ENTRIES IN ICOUNT?
395C
396  234    IF(ISECRI.EQ.1) GO TO 235
397         IF((ICTNXT-1).GE.(64*(INOTBL-1)+ICTLST)) GO TO 235
398         ISECRI = ISECRI/2
399         ICTNXT = ILSTNG + 1
400         IF(ISECRI.LT.1) ISECRI = 1
```

```
401         GO TO 205
402C
403C        PRINT FIXED AND VARIABLE STRINGS
404C
405   235   CONTINUE
406C        WRITE(6,10) ((I-1,FFD(I,66)),I=1,63),FFD(64,66),
407C    &    FFD(65,66),IEOR,IBLKSV
408C        WRITE(6,16)
409C        WRITE(6,17)
410C        WRITE(6,17) ((ICOUNT(IY,IX),IX=1,8),IY=2,ICTLST)
411C        WRITE(6,17)
412C        WRITE(6,17) ((ICOUNT(IY,IX),IX=1,8),IY=ICTLST+1
413C    &         ILSTNG)
414C        WRITE(6,17)
415C        WRITE(6,17) ((ICOUNT(IY,IX),IX=1,8),IY=ILSTNG+1,
416C    &         ICTNXT-1)
417C        WRITE(6,17)
418C        WRITE(6,17)
419C
420C         SORT REPEATED AND FIXED STRINGS BY WEIGHT
421C
422   236   ICOUNT(1,1) = 0
423         ICOUNT(1,3) = 0
424         ICOUNT(1,8) = 0
425         DO 237 ICT=4,7
426   237   ICOUNT(1,ICT) = IBLANK
427         DO 242 ICT=1,ICTNXT-1
428         DO 240 JCT=1,8
429   240   IRANK(ICT,JCT) = ICOUNT(ICT,JCT)
430   242   CONTINUE
431         DO 260 ISORT=2,ICTNXT-1
432         ILOC = ISORT
433         IMAX = IRANK(ISORT,2)
434         IF((ICTNXT-1).EQ.ISORT) GO TO 245
435         DO 244 JSORT = ISORT+1,ICTNXT-1
436         IF(IRANK(JSORT,2).LE.IMAX) GO TO 244
437         ILOC = JSORT
438         IMAX = IRANK(JSORT,2)
439   244   CONTINUE
440   245   DO 250 ICT = 1,8
441         IRANK(ISORT-1,ICT) = IRANK(ILOC,ICT)
442         IF(ILOC.NE.ISORT) IRANK(ILOC,ICT)=IRANK(ISORT,ICT)
443   250   CONTINUE
444   260   CONTINUE
445C
446C         PROCESS SORTED STRINGS
447C
448         IPROFL = 0
449         ILAST = IRANK(63*(INOTBL-1)-2,2)
450         IF(ILAST.EQ.0) ILAST = 1
```

```
451  270    DO 300 IPROC=1,ICTNXT-2
452         IF(ICOUNT(IPROC,1).EQ.3) GO TO 300
453         IF(ICOUNT(IPROC,1).EQ.4) GO TO 300
454         IF(ICOUNT(IPROC,2).LT.ILAST) GO TO 300
455         IPROFL = IPROFL + 1
456         ICOUNT(IPROC,1) = 4
457         DO 280 ICT=1,4
458  280    ISTRNG(ICT) = ICOUNT(IPROC,3+ICT)
459         DECODE(CSTRNG,15) (ICST(1,IX),IX=1,24)
460         DO 290  ICT=1,ICOUNT(IPROC,3)
461         IF(ICT.NE.1) GO TO 284
462         IREP = 0
463         GO TO 288
464  284    IF(ICST(1,ICT-1).EQ.ICST(1,ICT)) GO TO 288
465         IF(IREP.GT.1) IRD(ICST(1,ICT-1)+1,IREP) =
466     &    IRD(ICST(1,ICT-1)+1,IREP) - ICOUNT(IPROC,8)
467         IREP = 0
468  288    IREP = IREP + 1
469         IX = ICST(1,ICT) + 1
470         FFD(IX,66) = FFD(IX,66) - ICOUNT(IPROC,8)
471  290    FFD(65,66) = FFD(65,66) - ICOUNT(IPROC,8)
472         IF(IREP.GT.1) IRD(ICST(1,ICT)+1,IREP) =
473     &    IRD(ICST(1,ICT)+1,IREP) - ICOUNT(IPROC,8)
474  300    CONTINUE
475         ISELEN = ISELEN + IPROFL
476C        WRITE(6,22) IPROFL, ISELEN
477         IF(IPROFL.NE.0) GO TO 202
478C
479C       ADJUST FFD FOR SELECTED DUPLICATE STRINGS
480C
481         IF((ICTNXT-2).LE.(63*(INOTBL-1)-2)) IAD=ICTNXT-2
482         IF((ICTNXT-2).GT.(63*INOTBL-65)) IAD=63*(INOTBL-1)-2
483         DO 310 ICT = 1,IAD
484         IF(IRANK(ICT,1).NE.3) GO TO 310
485         IF(IRANK(ICT,2).LT.ILAST) GO TO 310
486         INUM = FLD(0,6,IRANK(ICT,4))
487         FFD(INUM+1,66) = FFD(INUM+1,66)-IRANK(ICT,3)*
488     &        IRANK(ICT,8)
489         FFD(65,66) = FFD(65,66) - IRANK(ICT,3)*IRANK(ICT,8)
490         IRD(INUM+1,IRANK(ICT,3)) = 0
491  310    CONTINUE
492C
493C       ADJUST FFD FOR UNSELECTED DUPLICATE STRINGS
494C
495         DO 330  JCT=4,63
496         DO 320  ICT=1,64
497         IF(JCT.EQ.4) IRD(ICT,1) = 0
498         IRPC = IRPC + IRD(ICT,JCT)
499         FFD(JCT+1,66) = FFD(JCT+1,66) + IRD(ICT,JCT)
500         FFD(ICT,66) = FFD(ICT,66) - (JCT-1)*IRD(ICT,JCT)
```

```
501           FFD(65,66) = FFD(65,66) - (JCT-2)*IRD(ICT,JCT)
502     320   CONTINUE
503     330   CONTINUE
504           DO 350  JCT=64,80
505           DO 340  ICT=1,64
506           IRPC = IRPC + IRD(ICT,JCT)
507           FFD(1,66) = FFD(1,66) + IRD(ICT,JCT)
508           FFD(JCT-63,66) = FFD(JCT-63,66) + IRD(ICT,JCT)
509           FFD(ICT,66) = FFD(ICT,66) - (JCT-1)*IRD(ICT,JCT)
510           FFD(65,66) = FFD(65,66) - (JCT-3)*IRD(ICT,JCT)
511     340   CONTINUE
512     350   CONTINUE
513C
514C          WRITE FFD AND ICOUNT TABLES
515C
516C          WRITE(6,10)((I-1,FFD(I,66)),I=1,63),FFD(64,66),
517C         &  FFD(65,66),IEOR,IBLKSV
518C          WRITE(6,17)
519C          WRITE(6,17) (((IRANK(IY,IX),IX=1,8),IY=1,IADJLT))
520C          WRITE(6,23) IRPC
521C
522C         FILL IRANK WITH SELECTED STRINGS(4),
523C         ORIGINAL SYMBOLS (5), NO. OF END OF
524C         RECORD CHARACTER'S(6), AND NO. OF UNREPLACED
525C         REPEATED STRINGS(7)
526C
527           DO 360 ICT=1,8
528     360   IRANK(IADJLT+1,ICT)=IRANK(1,ICT)
529           IADJLT=IADJLT+1
530           ISHIFT=64*64*64*64*64
531           DO 370 JCT=1,64
532           IRANK(IADJLT+JCT,1)=5
533           IRANK(IADJLT+JCT,2)=FFD(JCT,66)
534           IRANK(IADJLT+JCT,3)=1
535           IRANK(IADJLT+JCT,4)=IZERBK+(JCT-1)*ISHIFT
536           IRANK(IADJLT+JCT,5)=IBLANK
537           IRANK(IADJLT+JCT,6)=IBLANK
538           IRANK(IADJLT+JCT,7)=IBLANK
539     370   IRANK(IADJLT+JCT,8)=FFD(JCT,66)
540           IADJLT=IADJLT+64
541           DO 390 JCT=1,2
542           IRANK(IADJLT+JCT,1)=JCT+5
543           IF(JCT.EQ.1)IRANK(IADJLT+JCT,2)=IEOR
544           IF(JCT.EQ.2)IRANK(IADJLT+JCT,2)=IRPC
545           IRANK(IADJLT+JCT,3)=0
546           IF(JCT.EQ.1)IRANK(IADJLT+JCT,4)=LEOR
547           IF(JCT.EQ.2)IRANK(IADJLT+JCT,4)=LREP
548           DO 380 ICT=5,7
549     380   IRANK(IADJLT+JCT,ICT)=IBLANK
550           IF(JCT.EQ.1)IRANK(IADJLT+JCT,8)=IEOR
```

```
551         IF(JCT.EQ.2)IRANK(IADJLT+JCT,8)=IRPC
552   390   CONTINUE
553         IADJLT=IADJLT+2
554          ISORFL = 1
555C
556C         SORT SELECTED STRINGS, ORIGINAL
557C         SYMBOLS,END OF RECORD, AND UNREPLACED
558C         CHARACTER STRINGS
559C
560   395    DO 416 ISORT = 2,IADJLT
561         ILOC=ISORT
562         IMAX=IRANK(ISORT,2)
563         IF(IADJLT.EQ.ISORT)GO TO 410
564         DO 400 JSORT=ISORT+1,IADJLT
565         IF(IRANK(JSORT,2).LE.IMAX)GO TO 400
566         ILOC=JSORT
567         IMAX=IRANK(JSORT,2)
568   400   CONTINUE
569   410   DO 415 ICT=1,8
570         IRANK(ISORT-1,ICT)=IRANK(ILOC,ICT)
571         IF(ILOC.NE.ISORT)IRANK(ILOC,ICT)=IRANK(ISORT,ICT)
572   415   CONTINUE
573   416   CONTINUE
574          GO TO (418,440),ISORFL
575C
576C         DETERMINE IF DUPLICATE ENTRIES OF
577C         LENGTH 2 BELONG IN TABLE 1 (INITIAL)
578C
579   418    IENSEL = 0
580         IADLST = IADJLT - 1
581         ISORFL = 2
582         IDSEL = IRANK(65-INOTBL,2)
583         DO 420 ICT=1,64
584         IF(IRD(ICT,2).LE.IDSEL) GO TO 420
585         IENSEL = IENSEL + 1
586         IRANK(IADJLT,1) = 3
587         IRANK(IADJLT,2) = IRD(ICT,2)
588         IRANK(IADJLT,3) = 2
589         ENCODE(CSTRNG,15) ICT-1,ICT-1,(IBLANK,I=1,22)
590         IRANK(IADJLT,4) = ISTRNG(1)
591         IRANK(IADJLT,5) = ISTRNG(2)
592         IRANK(IADJLT,6) = ISTRNG(3)
593         IRANK(IADJLT,7) = ISTRNG(4)
594         IRANK(IADJLT,8) = IRD(ICT,2)
595         ICOUNT(IENSEL+1,2) = IRD(ICT,2)
596         IADJLT = IADJLT + 1
597   420   CONTINUE
598         IF(IENSEL.EQ.0) GO TO 440
599C
600C         DETERMINE IF DUPLICATE ENTRIES OF
```

```
601C        LENGTH 2 BELONG IN TABLE 1 (FINAL)
602C
603         DO 423 ISORT = 2,IENSEL + 1
604         ILOC = ISORT
605         IMAX = ICOUNT(ISORT,2)
606         IF(ISORT.EQ.IENSEL+1) GO TO 422
607         DO 421 JSORT=ISORT+1,IENSEL+1
608         IF(ICOUNT(JSORT,2).LE.IMAX) GO TO 421
609         ILOC = JSORT
610         IMAX = ICOUNT(JSORT,2)
611   421   CONTINUE
612   422   ICOUNT(ISORT-1,2) = ICOUNT(ILOC,2)
613         IF(ILOC.NE.ISORT) ICOUNT(ILOC,2) = ICOUNT(ISORT,2)
614   423   CONTINUE
615         ISTART = 1
616         IENUSD = IENSEL
617         DO 431 JCT=1,IENSEL
618         DO 425 ICT=ISTART,IADLST
619         IF(IRANK(IADLST-ICT+1,1).EQ.3) GO TO 426
620   425   CONTINUE
621         GO TO 431
622   426   IREF = IADLST - ICT + 1
623         ICHK = 66 - INOTBL - JCT
624         IF(ICOUNT(JCT,2).GT.IRANK(ICHK,2)) GO TO 427
625         IDSEL = IRANK(ICHK,2)
626         IENUSD = JCT - 1
627         GO TO 432
628   427   ISTART = ICT + 1
629         IRANK(IREF,2) = 0
630C
631C         REMOVE EFFECT IN CHANGE IN ENTRY SELECTION
632C
633         ICHAR = FLD(0,6,IRANK(IREF,4))
634         IF(IRANK(IREF,3).GT.3) GO TO 428
635         IRD(ICHAR+1,1) = IRD(ICHAR+1,1) +
636        &   IRANK(IREF,3)* IRANK(IREF,8)
637         GO TO 431
638   428   IRPC = IRPC + IRANK(IREF,8)
639         IF(IRANK(IREF,3).GT.63) GO TO 429
640         IRD(IRANK(IREF,3)+1,1) = IRD(IRANK(IREF,3)+1,1)
641        &  + IRANK(IREF,8)
642         GO TO 430
643   429   IRD(1,1) = IRANK(IREF,8)
644         IREM = IRANK(IREF,3) - 64
645         IRD(IREM+1,1) = IRD(IREM+1,1) + IRANK(IREF,8)
646   430   IRD(ICHAR+1,1) = IRD(ICHAR+1,1) + IRANK(IREF,8)
647   431   CONTINUE
648   432   DO 434 ICT=1,IENSEL
649         IF(IRANK(IADLST+ICT,2).LE.IDSEL) GO TO 433
650         ICHAR = FLD(0,6,IRANK(IADLST+ICT,4))
```

```
651          IRD(ICHAR+1,1) = IRD(ICHAR+1,1) -
652         &   2*IRANK(IADLST+ICT,8)
653          GO TO 434
654   433    IRANK(IADLST+ICT,2) = 0
655   434    CONTINUE
656   435    DO 437 ICT=1,IADJLT
657          IF(IRANK(ICT,1).NE.5) GO TO 436
658          ICHAR = FLD(0,6,IRANK(ICT,4))
659          IRANK(ICT,2) =  IRANK(ICT,2) + IRD(ICHAR+1,1)
660          IRANK(ICT,8) = IRANK(ICT,2)
661          GO TO 437
662   436    IF(IRANK(ICT,1).NE.6) GO TO 437
663          IRANK(ICT,2) = IEOR
664          IRANK(ICT,8) = IEOR
665   437    CONTINUE
666          DO 439 ICT=1,8
667   439    IRANK(IADJLT,ICT) = IRANK(1,ICT)
668          GO TO 395
669   440    IADJLT = IADLST
670C
671C         WRITE IRANK TO TERMINAL
672C
673         WRITE(6,36)
674         WRITE(6,26)
675         WRITE(6,16)
676         WRITE(6,26)
677         WRITE(6,17)((IRANK(IY,IX),IX=1,8),IY=1,IADJLT)
678         WRITE(6,26)
679C
680C         ASSIGN SYMBOLS AND STRINGS TO
681C         "PROPER" TABLES
682C
683         DO 444 ICT=1,64*INOTBL
684         ICOUNT(ICT,1) = 0
685         ICOUNT(ICT,2) = 0
686         ICOUNT(ICT,3) = 0
687         DO 442 JCT=4,7
688   442   ICOUNT(ICT,JCT) = IBLANK
689   444   ICOUNT(ICT,8) = 0
690C
691C         ASSIGN TABLE 1
692C
693         DO 450 JCT=1,65-INOTBL
694         IF(IRANK(JCT,1).NE.5)GO TO 450
695         INUM=FLD(0,6,IRANK(JCT,4))
696         DO 448 ICT=1,8
697   448   ICOUNT(INUM+1,ICT)=IRANK(JCT,ICT)
698   450   CONTINUE
699         IPOS=1
700         DO 490 JCT=1,65-INOTBL
```

```
      IF(IRANK(JCT,1).EQ.5)GO TO 490
460   IF(ICOUNT(IPOS,1).EQ.0) GO TO 470
      IPOS=IPOS+1
      GO TO 460
470   DO 480 ICT=1,8
480   ICOUNT(IPOS,ICT)=IRANK(JCT,ICT)
      IPOS=IPOS+1
490   CONTINUE
      DO 520 JCT=2,INOTBL
500   IF(ICOUNT(IPOS,1).EQ.0)GO TO 510
      IPOS=IPOS+1
      GO TO 500
510   ITAFLG(JCT)=IPOS
      ICOUNT(IPOS,1)=8
      ICOUNT(IPOS,2)=0
      ICOUNT(IPOS,3)=0
      ICOUNT(IPOS,4)=ITIF1
      ICOUNT(IPOS,5)=ITIF2
      ICOUNT(IPOS,6)=IZERBK+JCT*ISHIFT
      ICOUNT(IPOS,7)=IBLANK
      ICOUNT(IPOS,8)=0
520   CONTINUE
C
C     WRITE TABLE 1 TO TERMINAL
C
       WRITE(6,33)
       WRITE(6,27)
       WRITE(6,17)
      WRITE(6,38)(IY-1,(ICOUNT(IY,IX),IX=1,8),IY=1,64)
C
C     ASSIGN TABLE 2
C
C     ASSIGN REST OF ALPHABET FIRST
C
       DO 540 JCT=66-INOTBL,64*INOTBL-(INOTBL-1)
      IF(IRANK(JCT,1).NE.5)GO TO 540
      INUM=FLD(0,6,IRANK(JCT,4))
      DO 530 ICT=1,8
530   ICOUNT(INUM+65,ICT)=IRANK(JCT,ICT)
540   CONTINUE
C
C     COMPLETE TABLE
C
       IPOS = 65
       DO 580 JCT=66-INOTBL,64*INOTBL-(INOTBL-1)
      IF(IRANK(JCT,1).EQ.0) GO TO 580
      IF(IRANK(JCT,1).EQ.5)GO TO 580
550   IF(ICOUNT(IPOS,1).EQ.0)GO TO 560
      IPOS=IPOS+1
C
```

```
C       CHECK TO SEE IF SECOND TABLE FULL
C
        IF(IPOS.GT.128)GO TO 590
        GO TO 550
  560   DO 570 ICT=1,8
  570   ICOUNT(IPOS,ICT)=IRANK(JCT,ICT)
        IPOS=IPOS+1
        IF(IPOS.GT.128)GO TO 590
  580   CONTINUE
  590   CONTINUE
C
C       WRITE TABLE 2 TO TERMINAL
C
         WRITE(6,28)
         WRITE(6,17)
        WRITE(6,38)(IY-1,(ICOUNT(IY+64,IX),IX=1,8),IY=1,64)
        IF(INOTBL.EQ.2)GO TO 800
C
C       ASSIGN OTHER TABLES
C
        DO 630  KCT=3,INOTBL
        IPOS=64*(KCT-1)+1
        DO 610 JCT=JCT,64*INOTBL-(INOTBL-1)
        IF(IRANK(JCT,1).EQ.5)GO TO 610
        DO 600 ICT=1,8
  600   ICOUNT(IPOS,ICT)=IRANK(JCT,ICT)
        IPOS=IPOS+1
        IF(IPOS.GT.64*KCT)GO TO 620
  610   CONTINUE
  620   CONTINUE
C
C       WRITE TABLE TO TERMINAL
C
         WRITE(6,29) KCT
         WRITE(6,17)
        WRITE(6,38) (IY-1,(ICOUNT(IY+64*(KCT-1),IX),IX=1,8)
     &    ,IY=1,64)
  630   CONTINUE
C
C       CALCULATE NO OF ENTRIES IN TABLES 2-N
C
  800   DO 820 JCT=2,INOTBL
        DO 810 ICT=1,64
        ICOUNT(ITAFLG(JCT),2)=ICOUNT(ITAFLG(JCT),2)+
     & (ICOUNT(64*(JCT-1)+ICT,8))
  810   CONTINUE
        ICOUNT(ITAFLG(JCT),8)=ICOUNT(ITAFLG(JCT),2)
  820   CONTINUE
C
C        WRITE NO OF ENTRIES IN TABLES 2-N TO TERMINAL
```

```
801C
802         WRITE(6,24)(ICT,ICOUNT(ITAFLG(ICT),2),ICT=2,INOTBL)
803C
804C        COUNT ENTRIES IN RESPECTIVE POSITIONS
805C
806         DO 830 JCT=1,65
807   830   FFD(JCT,66)=0
808         DO 850 JCT=1,64
809         DO 840 ICT=1,INOTBL
810         FFD(JCT,66)=FFD(JCT,66)+ICOUNT(64*(ICT-1)+JCT,8)
811   840   CONTINUE
812         FFD(65,66)=FFD(65,66)+FFD(JCT,66)
813   850   CONTINUE
814C
815C        WRITE FFD(I,66)TABLE
816C
817          WRITE(6,34)
818          WRITE(6,17)
819         WRITE(6,10)((I-1,FFD(I,66)),I=1,63),FFD(64,66),
820       & FFD(65,66),IEOR,IBLKSV
821C
822C          OUTPUT ENCODING/DECODING TABLES
823C
824          WRITE(40,17) ((ICOUNT(ICT,JCT),JCT=1,8),ICT=1,
825       &       INOTBL*64)
826C
827C        CALCULATE ENTROPY,HC0
828C
829          HC0S = 0
830         DO 860 ICT=1,64
831         IF(FFD(ICT,66).EQ.0)GO TO 860
832         HC0S=HC0S+FFD(ICT,66)*ALOG(FFD(65,66)/FFD(ICT,66))
833   860   CONTINUE
834         HC0=HC0S/ALOG(2)/FFD(65,66)
835C
836C        WRITE ENTROPY WITH STRINGS SUBSTITUTED
837C
838         WRITE (6,19)HC0
839          BITAVG = 6. * FFD(65,66)/ITOTIN
840          WRITE(6,25) BITAVG
841          WRITE(6,26)
842C
843C         TAIL
844C
845   990    CALL DATIM(IDATE,TIME)
846          PRINT ," "
847          PRINT ," "
848          PRINT 9, IDATE,TIME
849          PRINT ," "
850          PRINT ,"PROGRAM END"
```

```
851         STOP
852         END
```

```
1       SYMDEF  REFORM          REFORM PROGRAM
2*
3*  REFORMATS CARD IMAGE
4*   CALL(INPUT,LLSIZ)
5*
6*  INPUT - 80 WORD AREA FOR STORING CARD IMAGE (1CHAR./WORD)
7*  LLSIZ - CURRENT SIZE OF FILE
8*
9REFORM SREG     REGF       SAVES FORTRAN REGISTERS
10        LDA      SFLAG      FLAG DETERMINES SEARCH ENTRY NO.
11        CMPA     0,DU
12        TNZ      LNG03-1    BRANCH IF NOT FIRST ENTRY
13        AOS      SFLAG
14        LXL4     3,1*       X4 =) CURRENT SIZE OF FILE
15        STZ      LLINK      LLINK =) LLINK BEING PROCESSED
16        STZ      IMANO      IMANO =) CARD POSITION WITHIN LL.
17USEDL   EAQ      NBLKS      BUILDS READ DCW
18        SBLX4    NBLKS,DU
19        TPL      *+4
20        EAQ      NBLKS,4
21        TZE      RETFL      FINISHED
22        TMI      RETFL      FINISHED
23        MPY      320,DU
24        ORA      IOBUF,DU   A =) READ DCW
25        EAX2     IOBUF,AL   X2 =) WORD COUNT IN BUFFER
26        STX2     LNG04+3    LNG04+2 =) WORD COUNT
27        TRA      DSCINP     BRANCH TO READ; RETUL =) RETURN
28RETUL   EAX3     IOBUF      X3 =) BUFFER ADDRESS
29        LDA      0,3        BLOCK CONTROL WORD
30        EAX2     ,AL        X2=) # OF WORDS USED
31        TZE      FSERR      ERROR RETURN(NO WORDS USED)
32        CMPX2    320,DU
33        TRC      FSERR      ERROR RETURN(TOO MANY WORD'S USED)
34        ARL      18
35        SBA      1,DL
36        CMPA     LLINK
37        TNZ      FSERR      ERROR RETURN(BLOCK NO.NE.LLINK NO.)
38        STX3     *+1
39        EAX2     **,2       X2=) BUFFER ADDRESS + NO. WORDS USED
40        STX2     LNG03      LNG03 =) LAST WORD USED(LOCATION)
41        EAX2     ,3         X2 =) BUFFER ADDRESS
42        TRA      *+2
43        LREG     REGG       RESTORE GMAP REGISTERS
44LNG03   CMPX2    **,DU
45        TZE      LNG04      TRA IF AT END OF BLOCK
46        TRC      FSERR      ERR. RETURN(RECORD SIZE ) BLOCK SIZE)
47        ADLX2    1,DU
48        LDA      0,2        RECORD CONTROL WORD
49        CANA     -1,DU
50        TZE      RETFL      NORMAL RETURN(END-OF-FILE)
```

```
51          ADLX2   0,2         X2 =) ADVANCE TO NEXT LOGICAL RECORD)
52          ANA     -1,DU
53          CMPA    14,DU
54          TNZ     FSERR       ERROR RETURN(NOT 14 WORDS/RECORD)
55          AOS     IMANO       INCREMENT CARD COUNT
56          EAA     -13,2       PREPARE TALLY'S
57          ASA     TAL1        STORE TALLY FOR BUFFER INPUT
58          LDA     3,1         CARD IMAGE CHARACTER BUFFER
59          ASA     TAL2        STORE TALLY FOR OUTPUT BUFFER
60          LDA     TAL1,SC
61          STA     TAL2,ID
62          TTF     *-3
63          LDAQ    TAL1+2      TALLY REFRESH
64          STAQ    TAL1
65          SREG    REGG        SAVE GMAP REGISTERS
66          LREG    REGF        RESTORE FORTRAN REGISTERS
67          TRA     0,1         RETURN TO PROCESS TAPE CARD
68LNG04      ADLX3   320,DU      ADVANCE TO NEXT LLINK IN BUFFER
69          AOS     LLINK
70          STZ     IMANO       ZEROS CARD COUNT
71          CMPX3   **,DU
72          TNC     RETUL+1     CONTINUE PROCESSING BUFFER
73          TRA     USEDL       TRA TO READ MORE LLINK'S
74DSCINP     STA     LDCW2       A =) DATA DCW
75          LDQ     =5,DL
76          MPY     LLINK
77          STQ     LDCW2+1     Q =) SECTOR NUMBER
78          MME     GEINOS      READS ONLY 1 LLINK
79          SDIA
80          ZERO    LALTNA,LDCW1
81          RDIC
82          ZERO    LALTNA,LDCW2
83          ZERO    ISTAT
84          MME     GEROAD      WAIT FOR I/O
85          LDA     ISTAT       CHECK FOR STATUS
86          ANA     =0370000,DU
87          TZE     RETUL       TRA ON SUCCESSFUL READ
88          LDA     ISTAT
89          STA     3,1*        STORES STATUS IN LLSIZ
90          TRA     RETFL+1
91FSERR     ORA     -1,DU       FILE SYSTEM ERROR
92          STA     3,1*
93          TRA     RETFL+1
94RETFL      STZ     3,1
95          STZ     SFLAG       RESET ENTRY FLAG
96          LREG    REGF        LOAD FORTRAN REGISTERS
97          TRA     0,1         RETURN TO FORTRAN PROGRAM
98*
99*          DATA WORDS
100*
```

```
101IMANO  OCT     0          CARD IMAGE NUMBER WITHIN LLINK
102LALTNA OCT     200        FC 20
103ISTAT  BSS     2          STATUS WORDS
104LDCW1  IOTD    LDCW2+1,1
105LDCW2 EBSS     2          DCW2(L(BUF),WD. CT; SECTOR NUMBER)
106LLINK  OCT     0          CURRENT LLINK BEING PROCESSED
107LLSIZ  OCT     0          CURRENT NUMBER OF LLINK'S
108NBLKS   EQU     1          ONLY 1 LLINK PER I/O
109REGF   8BSS     8          FORTRAN REGISTERS
110REGG   8BSS     8          GMAP REGISTERS
111SFLAG  OCT     0          SEARCH FLAG
112TAL1   ETALLY  0,80
113TAL2    TALLY  0,80
114       TALLY  0,80
115       TALLY  0,80
116TEMP   OCT     0          TEMPORY STORAGE
117IOBUF EBSS     NBLKS*320  INPUT BUFFER AREA
118FMSBUF EQU     IOBUF      FMS BUFFER AREA
119       END
```

```
1C        JCL ENCODE PROGRAM
2C        JCL VARIABLES (572 WORDS)
3C
4         DIMENSION ISYNM(6,8),IS1(32,5),IS2(5,5),ISYCH(6,24)
5         DIMENSION IA(64),IDUPE(20,3),ITABLE(3),IDUPS(64)
6C                                                    ,
7         DATA IEALPH/64/
8         DATA IEDUPE/20/
9         DATA IESLST/ 6/
10         DATA IETBLE/ 2/
11C
12C        ENCODING DATA TABLES
13C
14          DATA (ISYNM( 1,K),K=1,8)/  1,  1, 15,
15       &   6H$      ,6H #####,6H##     ,6H       , 32/
16C
17          DATA (IS1( 1,K),K=1,5)/0,6H FILE ,2*6H       ,  9/
18          DATA (IS1( 2,K),K=1,5)/0,6H TAPE9,2*6H       , 10/
19          DATA (IS1( 3,K),K=1,5)/0,6H LIMIT,6HS      ,0, 11/
20          DATA (IS1( 4,K),K=1,5)/0,6H IDENT,2*6H       , 12/
21          DATA (IS1( 5,K),K=1,5)/0,6H PRMFL,2*6H       , 13/
22          DATA (IS1( 6,K),K=1,5)/0,6H SELEC,6HT      ,0, 14/
23          DATA (IS1( 7,K),K=1,5)/0,6H EXECU,6HTE     ,0, 15/
24          DATA (IS1( 8,K),K=1,5)/0,6H IF    ,2*6H      , 23/
25          DATA (IS1( 9,K),K=1,5)/0,6H PROGR,6HAM     ,0, 26/
26          DATA (IS1(10,K),K=1,5)/0,6H LOWLO,6HAD     ,0, 32/
27          DATA (IS1(11,K),K=1,5)/0,6H ENDJO,6HB      ,0, 40/
28          DATA (IS1(12,K),K=1,5)/0,6H USE   ,2*6H      , 45/
29          DATA (IS1(13,K),K=1,5)/0,6H READ  ,2*6H      , 48/
30          DATA (IS1(14,K),K=1,5)/0,6H OPTIO,6HN      ,0, 53/
31          DATA (IS1(15,K),K=1,5)/0,6H WHEN  ,2*6H      , 54/
32          DATA (IS1(16,K),K=1,5)/0,6H INPUT,2*6H       , 61/
33          DATA (IS1(17,K),K=1,5)/0,6H CONVE,6HR      ,0, 64/
34          DATA (IS1(18,K),K=1,5)/0,6H SYSOU,6HT      ,0, 65/
35          DATA (IS1(19,K),K=1,5)/0,6H NOTE  ,2*6H      , 66/
36          DATA (IS1(20,K),K=1,5)/0,6H DATA  ,2*6H      , 67/
37          DATA (IS1(21,K),K=1,5)/0,6H MSG2  ,2*6H      , 69/
38          DATA (IS1(22,K),K=1,5)/0,6H GOTO  ,2*6H      , 70/
39          DATA (IS1(23,K),K=1,5)/0,6H LIMIT,2*6H       , 72/
40          DATA (IS1(24,K),K=1,5)/0,6H FUTIL,2*6H       , 83/
41          DATA (IS1(25,K),K=1,5)/0,6H 191PK,2*6H       , 89/
42          DATA (IS1(26,K),K=1,5)/0,6H LIBRA,6HRY     ,0, 99/
43          DATA (IS1(27,K),K=1,5)/0,6H PUNCH,2*6H       ,100/
44          DATA (IS1(28,K),K=1,5)/0,6H UTILI,6HTY     ,0,101/
45          DATA (IS1(29,K),K=1,5)/0,6H FORM  ,2*6H      ,102/
46          DATA (IS1(30,K),K=1,5)/0,6H TAPE7,2*6H       ,103/
47          DATA (IS1(31,K),K=1,5)/0,6H SET   ,2*6H      ,108/
48          DATA (IS1(32,K),K=1,5)/1H$,6H BREAK,2*6H       ,113/
49C
50C
```

```
51          DATA (ISYNM( 2,K),K=1,8)/  1,  1, 15,
52     &    6H%      ,6H #####,6H##     ,6H       ,  5/
53C
54          DATA (IS2( 1,K),K=1,5)/0,6H INFO ,2*6H       , 28/
55          DATA (IS2( 2,K),K=1,5)/0,6H JOBNA,6HME     ,0, 43/
56          DATA (IS2( 3,K),K=1,5)/0,6H WHEN ,2*6H       , 57/
57          DATA (IS2( 4,K),K=1,5)/0,6H IF   ,2*6H       , 58/
58          DATA (IS2( 5,K),K=1,5)/0,6H GOTO ,2*6H       , 81/
59C
60C
61          DATA (ISYNM( 3,K),K=1,8)/  2,  0,  4,
62     &    6HNNSP   ,3*6H        , 29/
63          DATA (ISYNM( 4,K),K=1,8)/  2,  0,  4,
64     &    6HNNST   ,3*6H        , 27/
65          DATA (ISYNM( 5,K),K=1,8)/  2,  0,  5,
66     &    6HDEN16  ,3*6H        , 24/
67          DATA (ISYNM( 6,K),K=1,8)/  2,  0, 18,
68     &    6H**,R,R,6H,ZZZOB,6HJ-NNS/,0, 56/
69C
70          DATA(IDUPS(K),K= 1, 8)/6H001003,6H000000,6H000000,
71     &    6H000000,6H000000,6H000000,6H000000,6H000000/
72          DATA(IDUPS(K),K= 9,16)/6H000000,6H000000,6H000000,
73     &    6H000000,6H000000,6H000000,6H000000,6H000000/
74          DATA(IDUPS(K),K=17,24)/6H00400@,6H000000,6H000000,
75     &    6H000000,6H000000,6H000000,6H000000,6H000000/
76          DATA(IDUPS(K),K=25,32)/6H000000,6H000000,6H000000,
77     &    6H000000,6H000000,6H000000,6H000000,6H000000/
78          DATA(IDUPS(K),K=33,40)/6H000000,6H000000,6H000000,
79     &    6H000000,6H000000,6H00:00:,6H000000,6H000000/
80          DATA(IDUPS(K),K=41,48)/6H000000,6H000000,6H000000,
81     &    6H000000,6H00)00A,6H000000,6H000000,6H000000/
82          DATA(IDUPS(K),K=49,56)/6H000000,6H000000,6H000000,
83     &    6H000000,6H000000,6H000000,6H000000,6H000000/
84          DATA(IDUPS(K),K=57,64)/6H000000,6H00B00B,6H000000,
85     &    6H00C00D,6H000000,6H000000,6H000000,6H000000/
86C
87          DATA (IDUPE( 1,J),J=1,3)/1H0, 2, 46/
88          DATA (IDUPE( 2,J),J=1,3)/1H0, 3, 98/
89          DATA (IDUPE( 3,J),J=1,3)/1H0, 4,119/
90          DATA (IDUPE( 4,J),J=1,3)/1H ,27, 71/
91          DATA (IDUPE( 5,J),J=1,3)/1H , 6, 80/
92          DATA (IDUPE( 6,J),J=1,3)/1H , 4, 82/
93          DATA (IDUPE( 7,J),J=1,3)/1H , 5, 84/
94          DATA (IDUPE( 8,J),J=1,3)/1H , 3, 85/
95          DATA (IDUPE( 9,J),J=1,3)/1H ,29, 97/
96          DATA (IDUPE(10,J),J=1,3)/1H ,28,105/
97          DATA (IDUPE(11,J),J=1,3)/1H ,26,114/
98          DATA (IDUPE(12,J),J=1,3)/1H ,15,116/
99          DATA (IDUPE(13,J),J=1,3)/1HN, 2, 60/
100          DATA (IDUPE(14,J),J=1,3)/1H*,67,106/
```

```
101           DATA (IDUPE(15,J),J=1,3)/1H*, 3, 86/
102           DATA (IDUPE(16,J),J=1,3)/1H*, 5,115/
103           DATA (IDUPE(17,J),J=1,3)/1H*,20,123/
104           DATA (IDUPE(18,J),J=1,3)/1HZ, 3, 62/
105           DATA (IDUPE(19,J),J=1,3)/1H,, 4, 47/
106           DATA (IDUPE(20,J),J=1,3)/1H,, 2, 30/
107         DATA (IA(K),K= 1, 8)/  0,  1,  2,  3,  4,  5,  6,  7/
108         DATA (IA(K),K= 9,16)/  8, 73, 74, 75, 76, 77, 78, 79/
109         DATA (IA(K),K=17,24)/ 16, 17, 18, 19, 20, 21, 22, 87/
110         DATA (IA(K),K=25,32)/ 88, 25, 90, 91, 92, 93, 94, 95/
111         DATA (IA(K),K=33,40)/ 96, 33, 34, 35, 36, 37, 38, 39/
112         DATA (IA(K),K=41,48)/104, 41, 42,107, 44,109,110,111/
113         DATA (IA(K),K=49,56)/112, 49, 50, 51, 52,117,118, 55/
114         DATA (IA(K),K=57,64)/120,121,122, 59,124,125,126,127/
115C
116           DATA IEOR/ 31/
117           DATA IREP/ 68/
118           DATA ITABLE/  0, 63,  0/
119C
120C         PROGRAM VARIABLES
121C
122           DIMENSION IDATE(2),INF(9),ISF(9),
123        &  INPUT(80),IOUT(100),ISIXD(6),ITEMP(4)
124           DIMENSION ISS(2)
125C
126           CHARACTER CNCDFN*12,CSRCFN*12,CSIX*6,CTEMP*24
127C
128           EQUIVALENCE (CNCDFN,INF(5)),(CSRCFN,ISF(5)),
129        &   (CSIX,ISIX),(CTEMP,ITEMP)
130           DATA IBLANK/16/
131           DATA IGOST/O 400000000000/
132           DATA IMOD/0/
133           DATA IPRMRD/1/
134           DATA IPRMWR/3/
135           DATA (ISF(I),I=1,4)/6HTCNJCL,2*6H      ,6H   /  /
136           DATA (ISF(I),I=7,9)/2*6H      ,6H"20" ;/
137           DATA (INF(I),I=1,4)/6HZZZSBO,6HDFILES,0,6H   /  /
138           DATA (INF(I),I=7,9)/2*6H      , 6H"30" ;/
139           DATA I20/20/
140           DATA I30/30/
141           DATA ISS/O1231241011124,O073040040040/
142        1  FORMAT (1X,A6,A2,2X,F6.3)
143        2  FORMAT (V)
144        3  FORMAT (80R1)
145        4  FORMAT (1X,"MATCH",3X,I3)
146        5  FORMAT (14A6)
147        6  FORMAT (1X,14A6)
148        7  FORMAT (1X,"INPUT = ",I6,2X,"BLKS SUPP. = ",
149         &  I6,2X,"OUTPUT = ",I6)
150C
```

```
151C        CONVERT GENERATED AND SPECIFIED
152C        STRINGS TO 1 SYMBOL/WORD
153C
154C
155C        HEAD
156C
157         CALL CALSUB(ISS)
158         CALL YNOSLU(30)
159         PRINT, "START ENCODE PROGRAM"
160         CALL DATIM (IDATE,TIME)
161         WRITE (6,1) IDATE,TIME
162         WRITE (6,1)
163         CALL FPARAM(1,80)
164         CALL FMEDIA(30,2)
165         DO 90 ICT=1, IESLST
166         DO 80 JCT=1,4
167 80      ITEMP(JCT) = ISYNM(ICT,JCT+3)
168 90      DECODE (CTEMP,3) (ISYCH(ICT,JCT), JCT=1,24)
169         IF(IREP.LT.64) IREPC = 2
170         IF(IREP.GE.64) IREPC = 3
171         IRCONT = 1+IREP/64
172         IRCHAR = IREP-64*(IRCONT-1)
173         IECONT = 1+IEOR/64
174         IECHAR = IEOR-64*(IECONT-1)
175C
176C         READ SOURCE AND ENCODE FILENAMES
177C         PRINT, " "
178         PRINT, "ENTER SOURCE FILE, ENCODED FILE"
179         READ (5,2) CSRCFN,CNCDFN
180C
181C         ATTACH INPUT SOURCE FILE
182C
183         CALL ATTACH (I20,ISF,IPRMRD,IMOD,ISTAT,)
184         IF((ISTAT.EQ.0).OR.(ISTAT.EQ.IGOST))GO TO 100
185         PRINT, "COULD NOT ATTACH INPUT FILE"
186         GO TO 999
187C
188C         ATTACH OUTPUT FILE
189C
190 100     CALL ATTACH (I30,INF,IPRMWR,IMOD,ISTAT,)
191         IF((ISTAT.EQ.0).OR.(ISTAT.EQ.IGOST)) GO TO 120
192         PRINT,"COULD NOT ATTACH OUTPUT FILE"
193         GO TO 999
194C         READ CARD IMAGE
195C
196C
197 120     READ(20,3,END=830) INPUT
198         IREC = IREC + 1
199         ICHIN = ICHIN + 80
200         IFEOR = 1
```

```
201C
202C        SUPPRESS TRAILING BLANKS
203C
204         DO 130 ICT=1,80
205         IF (INPUT(81-ICT).NE.IBLANK) GO TO 140
206 130     CONTINUE
207 140     ILAST = 81-ICT
208         IF (ILAST.EQ.80) GO TO 150
209         IBLKSV=IBLKSV+80-ILAST
210C
211C         PROCESS CARD IMAGE
212C
213         ILOOP = 1
214         IOUTPO = 1
215C
216C         CHECK FOR PRESENCE OF GENERATED
217C         AND SPECIFIED STRINGS
218C
219 150     DO 270 ICK= 1, IESLST
220         IF (ISYNM(ICK,2).EQ.0) GO TO 160
221         IF (ISYNM(ICK,2).NE.ILOOP) GO TO 270
222 160     DO 170  JCK=1,ISYNM(ICK,3)
223         IF (ISYCH(ICK,JCK).EQ.11) GO TO 170
224         IF (ISYCH(ICK,JCK).NE.INPUT
225      &  (ILOOP+JCK-1)) GO TO 270
226 170     CONTINUE
227         IF (ISYNM(ICK,1).NE.1) GO TO 290
228         ENCODE (CTEMP,3) (INPUT(KCT), KCT=ILOOP,ILOOP
229      &  + ISYNM(ICK,3)-1),(IBLANK,KCT = 1,24-ISYNM(ICK,3))
230         IPOCHK = ISYNM(ICK,3)/6
231         IREM = ISYNM(ICK,3)-6*IPOCHK
232         LSTART = 1
233         MSTART = 1
234 180     LCK=LSTART
235         IFLAG = 1
236         IF (ISYNM(ICK,3+LCK).NE.ITEMP(LCK)) GO TO 190
237         IF (LCK.EQ.IPOCHK) GO TO 230
238         LSTART = LSTART + 1
239         GO TO 180
240 190     DO 200 MCK=MSTART,ISYNM(ICK,8)
241            GO TO (192,194),ICK
242 192        IF (IS1(MCK,LCK).EQ.ITEMP(LCK)) GO TO 220
243            GO TO 200
244 194        IF (IS2(MCK,LCK).EQ.ITEMP(LCK)) GO TO 220
245         IF(IFLAG.EQ.1) GO TO 200
246         LSTART = 1
247         MSTART = MSTART + 1
248         IF(MSTART-ISYNM(ICK,8)) 180, 180, 270
249 200     CONTINUE
250         GO TO 270
```

```
251 220     IFLAG = 2
252         IF (LCK.EQ.IPOCHK) GO TO 230
253         LSTART = LSTART + 1
254         MSTART = MCK
255         GO TO 180
256 230     IF (IREM.EQ.0) GO TO 280
257            GO TO (232,234), ICK
258  232      ISIX = IS1(MCK,LCK+1)
259            GO TO 240
260 234        ISIX = IS2(MCK,LCK+1)
261 240     DECODE (CSIX,3) ISIXD
262         DO 250 NCK = 1, IREM
263         IF (INPUT(ILOOP+6*IPOCHK+NCK-1)
264      &  .NE.ISIXD(NCK)) GO TO 260
265 250     CONTINUE
266         GO TO 280
267 260     LSTART = 1
268         MSTART = MSTART + 1
269         IF (MSTART-ISYNM(ICK,8)) 180,180,270
270 270     CONTINUE
271C
272C         NO MATCH
273C
274         GO TO 310
275C
276C         MATCH
277C
278 280        GO TO (282,284),ICK
279 282        ICONT = IS1 (MCK,5)
280            GO TO 300
281 284        ICONT = IS2 (MCK,5)
282            GO TO 300
283 290     ICONT = ISYNM(ICK,8)
284 300     ILODEL = ISYNM (ICK,3)
285         IF((ILOOP.EQ.1).AND.(ISYNM(ICK,2).EQ.1)
286      &  .AND.(IREC.NE.1))IFEOR = 0
287         GO TO 800
288C
289C
290C         CHECK FOR DUPLICATE STRING
291C
292C
293 310     DO 320 ICK=ILOOP+1,ILAST
294         IF(INPUT(ILOOP).NE.INPUT(ICK)) GO TO 330
295 320     CONTINUE
296         ICK = ICK + 1
297 330     IRPCT=ICK-ILOOP
298         IF(IRPCT.EQ.1) GO TO 410
299C
300C         CHECK REPEAT ENTRIES IN EDT
```

```
301C
302         ISTART=FLD(0,18,IDUPS(INPUT(ILOOP)+1))
303         IF(ISTART.EQ.0) GO TO 350
304         IEND = FLD (18,18,IDUPS(INPUT(ILOOP)+1))
305         DO 340 ICK = ISTART, IEND
306         IF(IDUPE(ICK,2).EQ.IRPCT) GO TO 400
307 340     CONTINUE
308 350     IF(IA(INPUT(ILOOP)+1).LT.64) ICHREP = 1
309         IF(IA(INPUT(ILOOP)+1).GE.64)ICHREP = 2
310         IF((IREP*ICHREP).LE.(IREPC+ICHREP)) GO TO 410
311C
312C         GENERATE REPEAT STRING
313C
314         GO TO (370,360,360), IRCONT
315 360     IOUT(IOUTPO)=ITABLE(IRCONT)
316         IOUTPO = IOUTPO + 1
317 370     IOUT(IOUTPO) = IRCHAR
318         IOUTPO = IOUTPO + 1
319         IF(IRPCT.LT.64) ICTREP = 1
320         IF(IRPCT.GE.64) ICTREP = 2
321         GO TO (390,380), ICTREP
322 380     IOUT(IOUTPO) = 0
323         IOUTPO = IOUTPO +1
324         IOUT(IOUTPO) = IRPCT - 64
325         GO TO 395
326 390     IOUT(IOUTPO) = IRPCT
327  395    IOUTPO = IOUTPO + 1
328         ICONT = INPUT (ILOOP)
329         ILODEL = IRPCT
330         GO TO 800
331C
332C         PROCESS REPEAT ENTRY
333C
334 400     ICONT = IDUPE(ICK,3)
335         ILODEL = IRPCT
336         GO TO 800
337C
338C
339C         PROCESS ALPHABET
340 410     ICONT = IA(INPUT(ILOOP)+1)
341         ILODEL = 1
342         GO TO 800
343C         PLACE APPROPRIATE CHARACTERS IN BUFFER
344C
345 800     IOCONT = 1 + ICONT/64
346         ICHAR = ICONT-64*(IOCONT-1)
347         GO TO (820,810,810),IOCONT
348 810     IOUT(IOUTPO) = ITABLE(IOCONT)
349         IOUTPO = IOUTPO + 1
350 820     IOUT(IOUTPO) = ICHAR
```

```
351         IOUTPO = IOUTPO + 1
352         ILOOP = ILOOP + ILODEL
353C
354C         CHECK TO SEE IF INPUT
355C         BUFFER HAS BEEN PROCESSED
356C
357         IF (ILOOP.LE.ILAST) GO TO 150
358C
359C         WRITE INPUT & OUTPUT TO TERMINAL
360C
361         IF(IREC.EQ.1) GO TO 828
362         IF(IFEOR.EQ.0) GO TO 828
363         GO TO (824,822,822), IECONT
364 822     WRITE (30,3) ITABLE(IECONT), IECHAR
365         IOUTTO = IOUTTO + 2
366         IBLKSV = IBLKSV - 2
367         GO TO 828
368 824     WRITE (30,3) IECHAR
369         IOUTO = IOUTTO + 1
370         IBLKSV = IBLKSV - 1
371 828     IFEOR = 1
372         IUSED = IOUTPO - 1
373         WRITE (30,3) (IOUT(ICT), ICT = 1, IUSED)
374         IOUTTO = IOUTTO + IUSED
375         GO TO 120
376C
377C         OUTPUT SUMMARY STATISTICS
378C
379 830     GO TO (834,832,832), IECONT
380 832     WRITE (30,3) ITABLE (IECONT), IECHAR
381         IBLKSV = IBLKSV - 2
382         IOUTTO = IOUTTO + 2
383         GO TO 840
384 834     WRITE (30,3) IECHAR
385         IOUTTO = IOUTTO + 1
386         IBLKSV = IBLKSV - 1
387 840     END FILE 30
388         WRITE (6,7) ICHIN,ICHIN-IBLKSV,IOUTTO
389C
390C         TAIL
391C
392 999     PRINT, "STOP ENCODE PROGRAM"
393         CALL DATIM (IDATE,TIME)
394         WRITE (6,1) IDATE,TIME
395         CALL CALSUB(ISS)
396         STOP
397         END
```

```
1C        JCL DECODE PROGRAM
2C        JCL VARIABLES (896 WORDS USED, ABOUT 300 REQUIRED)
3C
4         DIMENSION IEDT(128,7),ITABLE(3)
5         DATA IEEDT/128/
6C
7C        DECODING DATA TABLES
8C
9          DATA (IEDT(  1,J),J=1,7)/ 5, 0, 1,
10      &   6H0      ,3*6H      /
11         DATA (IEDT(  2,J),J=1,7)/ 5, 0, 1,
12      &   6H1      ,3*6H      /
13         DATA (IEDT(  3,J),J=1,7)/ 5, 0, 1,
14      &   6H2      ,3*6H      /
15         DATA (IEDT(  4,J),J=1,7)/ 5, 0, 1,
16      &   6H3      ,3*6H      /
17         DATA (IEDT(  5,J),J=1,7)/ 5, 0, 1,
18      &   6H4      ,3*6H      /
19         DATA (IEDT(  6,J),J=1,7)/ 5, 0, 1,
20      &   6H5      ,3*6H      /
21         DATA (IEDT(  7,J),J=1,7)/ 5, 0, 1,
22      &   6H6      ,3*6H      /
23         DATA (IEDT(  8,J),J=1,7)/ 5, 0, 1,
24      &   6H7      ,3*6H      /
25         DATA (IEDT(  9,J),J=1,7)/ 5, 0, 1,
26      &   6H8      ,3*6H      /
27         DATA (IEDT( 10,J),J=1,7)/11, 1,15,
28      &   6H$      ,6H FILE ,2H*6H      /
29         DATA (IEDT( 11,J),J=1,7)/11, 1,15,
30      &   6H$      ,6H TAPE9,2H*6H      /
31         DATA (IEDT( 12,J),J=1,7)/11, 1,15,
32      &   6H$      ,6H LIMIT,6HS     ,6H      /
33         DATA (IEDT( 13,J),J=1,7)/11, 1,15,
34      &   6H$      ,6H IDENT,2H*6H      /
35         DATA (IEDT( 14,J),J=1,7)/11, 1,15,
36      &   6H$      ,6H PRMFL,2H*6H      /
37         DATA (IEDT( 15,J),J=1,7)/11, 1,15,
38      &   6H$      ,6H SELEC,6HT     ,6H      /
39         DATA (IEDT( 16,J),J=1,7)/11, 1,15,
40      &   6H$      ,6H EXECU,6HTE    ,6H      /
41         DATA (IEDT( 17,J),J=1,7)/ 5, 0, 1,
42      &   2H*6H       ,2H*6H      /
43         DATA (IEDT( 18,J),J=1,7)/ 5, 0, 1,
44      &   6HA      ,3*6H      /
45         DATA (IEDT( 19,J),J=1,7)/ 5, 0, 1,
46      &   6HB      ,3*6H      /
47         DATA (IEDT( 20,J),J=1,7)/ 5, 0, 1,
48      &   6HC      ,3*6H      /
49         DATA (IEDT( 21,J),J=1,7)/ 5, 0, 1,
50      &   6HD      ,3*6H      /
```

```
      DATA (IEDT( 22,J),J=1,7)/ 5, 0, 1,
     &   6HE      ,3*6H       /
      DATA (IEDT( 23,J),J=1,7)/ 5, 0, 1,
     &   6HF      ,3*6H       /
      DATA (IEDT( 24,J),J=1,7)/11, 1,15,
     &   6H$      ,6H IF    ,2H*6H       /
      DATA (IEDT( 25,J),J=1,7)/ 4, 0, 5,
     &   6HDEN16  ,3*6H       /
      DATA (IEDT( 26,J),J=1,7)/ 5, 0, 1,
     &   6HI      ,3*6H       /
      DATA (IEDT( 27,J),J=1,7)/11, 1,15,
     &   6H$      ,6H PROGR,6HAM     ,6H       /
      DATA (IEDT( 28,J),J=1,7)/ 4, 0, 4,
     &   6HNNST   ,3*6H       /
      DATA (IEDT( 29,J),J=1,7)/12, 1,15,
     &   6H%      ,6H INFO ,2H*6H       /
      DATA (IEDT( 30,J),J=1,7)/ 4, 0, 4,
     &   6HNNSP   ,3*6H       /
      DATA (IEDT( 31,J),J=1,7)/ 3, 0, 2,
     &   6H,,     ,3*6H       /
      DATA (IEDT( 32,J),J=1,7)/ 6, 6, 0,
     &   6HEOR    ,3*6H       /
      DATA (IEDT( 33,J),J=1,7)/11, 1,15,
     &   6H$      ,6H LOWLO,6HAD     ,6H       /
      DATA (IEDT( 34,J),J=1,7)/ 5, 0, 1,
     &   6HJ      ,3*6H       /
      DATA (IEDT( 35,J),J=1,7)/ 5, 0, 1,
     &   6HK      ,3*6H       /
      DATA (IEDT( 36,J),J=1,7)/ 5, 0, 1,
     &   6HL      ,3*6H       /
      DATA (IEDT( 37,J),J=1,7)/ 5, 0, 1,
     &   6HM      ,3*6H       /
      DATA (IEDT( 38,J),J=1,7)/ 5, 0, 1,
     &   6HN      ,3*6H       /
      DATA (IEDT( 39,J),J=1,7)/ 5, 0, 1,
     &   6HO      ,3*6H       /
      DATA (IEDT( 40,J),J=1,7)/ 5, 0, 1,
     &   6HP      ,3*6H       /
      DATA (IEDT( 41,J),J=1,7)/11, 1,15,
     &   6H$      ,6H ENDJO,6HB      ,6H       /
      DATA (IEDT( 42,J),J=1,7)/ 5, 0, 1,
     &   6HR      ,3*6H       /
      DATA (IEDT( 43,J),J=1,7)/ 5, 0, 1,
     &   6H-      ,3*6H       /
      DATA (IEDT( 44,J),J=1,7)/12, 1,15,
     &   6H%      ,6H JOBNA,6HME     ,6H       /
      DATA (IEDT( 45,J),J=1,7)/ 5, 0, 1,
     &   6H*      ,3*6H       /
      DATA (IEDT( 46,J),J=1,7)/11, 1,15,
     &   6H$      ,6H USE   ,2H*6H       /
```

```
      DATA (IEDT( 47,J),J=1,7)/ 3, 0, 2,
     &   6H00    ,3*6H      /
      DATA (IEDT( 48,J),J=1,7)/ 3, 0, 4,
     &   6H,,,,  ,3*6H      /
      DATA (IEDT( 49,J),J=1,7)/11, 1,15,
     &   6H$     ,6H READ ,2H*6H      /
      DATA (IEDT( 50,J),J=1,7)/ 5, 0, 1,
     &   6H/     ,3*6H      /
      DATA (IEDT( 51,J),J=1,7)/ 5, 0, 1,
     &   6HS     ,3*6H      /
      DATA (IEDT( 52,J),J=1,7)/ 5, 0, 1,
     &   6HT     ,3*6H      /
      DATA (IEDT( 53,J),J=1,7)/ 5, 0, 1,
     &   6HU     ,3*6H      /
      DATA (IEDT( 54,J),J=1,7)/11, 1,15,
     &   6H$     ,6H OPTIO,6HN     ,6H      /
      DATA (IEDT( 55,J),J=1,7)/11, 1,15,
     &   6H$     ,6H WHEN ,2H*6H      /
      DATA (IEDT( 56,J),J=1,7)/ 5, 0, 1,
     &   6HX     ,3*6H      /
      DATA (IEDT( 57,J),J=1,7)/ 4, 0,18,
     &   6H**,R,R,6H,ZZZOB,6HJ-NNS/,6H      /
      DATA (IEDT( 58,J),J=1,7)/12, 1,15,
     &   6H%     ,6H WHEN ,2H*6H      /
      DATA (IEDT( 59,J),J=1,7)/12, 1,15,
     &   6H%     ,6H IF   ,2H*6H      /
      DATA (IEDT( 60,J),J=1,7)/ 5, 0, 1,
     &   6H,     ,3*6H      /
      DATA (IEDT( 61,J),J=1,7)/ 3, 0, 2,
     &   6HNN    ,3*6H      /
      DATA (IEDT( 62,J),J=1,7)/11, 1,15,
     &   6H$     ,6H INPUT,2H*6H      /
      DATA (IEDT( 63,J),J=1,7)/ 3, 0, 3,
     &   6HZZZ   ,3*6H      /
      DATA (IEDT( 64,J),J=1,7)/ 8, 8, 0,
     &   6HTABLE ,6HSHIFT ,6H2     ,6H      /
      DATA (IEDT( 65,J),J=1,7)/11, 1,15,
     &   6H$     ,6H CONVE,6HR     ,6H      /
      DATA (IEDT( 66,J),J=1,7)/11, 1,15,
     &   6H$     ,6H SYSOU,6HT     ,6H      /
      DATA (IEDT( 67,J),J=1,7)/11, 1,15,
     &   6H$     ,6H NOTE ,2H*6H      /
      DATA (IEDT( 68,J),J=1,7)/11, 1,15,
     &   6H$     ,6H DATA ,2H*6H      /
      DATA (IEDT( 69,J),J=1,7)/ 7, 7, 0,
     &   6HREP   ,3*6H      /
      DATA (IEDT( 70,J),J=1,7)/11, 1,15,
     &   6H$     ,6H MSG2 ,2H*6H      /
      DATA (IEDT( 71,J),J=1,7)/11, 1,15,
     &   6H$     ,6H GOTO ,2H*6H      /
```

```
      DATA (IEDT( 72,J),J=1,7)/ 3, 0,27,
     &   2H*6H       ,2H*6H       /
      DATA (IEDT( 73,J),J=1,7)/11, 1,15,
     &   6H$      ,6H LIMIT,2H*6H      /
      DATA (IEDT( 74,J),J=1,7)/ 5, 0, 1,
     &   6H9      ,3*6H       /
      DATA (IEDT( 75,J),J=1,7)/ 5, 0, 1,
     &   6H(      ,3*6H       /
      DATA (IEDT( 76,J),J=1,7)/ 5, 0, 1,
     &   6H#      ,3*6H       /
      DATA (IEDT( 77,J),J=1,7)/ 5, 0, 1,
     &   6H@      ,3*6H       /
      DATA (IEDT( 78,J),J=1,7)/ 5, 0, 1,
     &   6H:      ,3*6H       /
      DATA (IEDT( 79,J),J=1,7)/ 5, 0, 1,
     &   6H)      ,3*6H       /
      DATA (IEDT( 80,J),J=1,7)/ 5, 0, 1,
     &   6H?      ,3*6H       /
      DATA (IEDT( 81,J),J=1,7)/ 3, 0, 6,
     &   2H*6H       ,2H*6H       /
      DATA (IEDT( 82,J),J=1,7)/12, 1,15,
     &   6H%      ,6H GOTO ,2H*6H      /
      DATA (IEDT( 83,J),J=1,7)/ 3, 0, 4,
     &   2H*6H       ,2H*6H       /
      DATA (IEDT( 84,J),J=1,7)/11, 1,15,
     &   6H$      ,6H FUTIL,2H*6H      /
      DATA (IEDT( 85,J),J=1,7)/ 3, 0, 5,
     &   2H*6H       ,2H*6H       /
      DATA (IEDT( 86,J),J=1,7)/ 3, 0, 3,
     &   2H*6H       ,2H*6H       /
      DATA (IEDT( 87,J),J=1,7)/ 3, 0, 3,
     &   6H***    ,3*6H       /
      DATA (IEDT( 88,J),J=1,7)/ 5, 0, 1,
     &   6HG      ,3*6H       /
      DATA (IEDT( 89,J),J=1,7)/ 5, 0, 1,
     &   6HH      ,3*6H       /
      DATA (IEDT( 90,J),J=1,7)/11, 1,15,
     &   6H$      ,6H 191PK,2H*6H      /
      DATA (IEDT( 91,J),J=1,7)/ 5, 0, 1,
     &   6H&      ,3*6H       /
      DATA (IEDT( 92,J),J=1,7)/ 5, 0, 1,
     &   6H.      ,3*6H       /
      DATA (IEDT( 93,J),J=1,7)/ 5, 0, 1,
     &   6H)      ,3*6H       /
      DATA (IEDT( 94,J),J=1,7)/ 5, 0, 1,
     &   6H(      ,3*6H       /
      DATA (IEDT( 95,J),J=1,7)/ 5, 0, 1,
     &   6H(      ,3*6H       /
      DATA (IEDT( 96,J),J=1,7)/ 5, 0, 1,
     &   6H-      ,3*6H       /
```

```
      DATA (IEDT( 97,J),J=1,7)/ 5, 0, 1,
     &  6H¢      ,3*6H      /
      DATA (IEDT( 98,J),J=1,7)/ 3, 0,29,
     &  2H*6H       ,2H*6H      /
      DATA (IEDT( 99,J),J=1,7)/ 3, 0, 3,
     &  6H000    ,3*6H      /
      DATA (IEDT(100,J),J=1,7)/11, 1,15,
     &  6H$      ,6H LIBRA,6HRY    ,6H       /
      DATA (IEDT(101,J),J=1,7)/11, 1,15,
     &  6H$      ,6H PUNCH,2H*6H       /
      DATA (IEDT(102,J),J=1,7)/11, 1,15,
     &  6H$      ,6H UTILI,6HTY    ,6H       /
      DATA (IEDT(103,J),J=1,7)/11, 1,15,
     &  6H$      ,6H FORM ,2H*6H       /
      DATA (IEDT(104,J),J=1,7)/11, 1,15,
     &  6H$      ,6H TAPE7,2H*6H       /
      DATA (IEDT(105,J),J=1,7)/ 5, 0, 1,
     &  6HQ      ,3*6H      /
      DATA (IEDT(106,J),J=1,7)/ 3, 0,28,
     &  2H*6H       ,2H*6H      /
      DATA (IEDT(107,J),J=1,7)/ 3, 0,67,
     &  6H******,6H******,6H******,6H******/
      DATA (IEDT(108,J),J=1,7)/ 5, 0, 1,
     &  6H$      ,3*6H      /
      DATA (IEDT(109,J),J=1,7)/11, 1,15,
     &  6H$      ,6H SET  ,2H*6H       /
      DATA (IEDT(110,J),J=1,7)/ 5, 0, 1,
     &  6H)      ,3*6H      /
      DATA (IEDT(111,J),J=1,7)/ 5, 0, 1,
     &  6H;      ,3*6H      /
      DATA (IEDT(112,J),J=1,7)/ 5, 0, 1,
     &  6H'      ,3*6H      /
      DATA (IEDT(113,J),J=1,7)/ 5, 0, 1,
     &  6H+      ,3*6H      /
      DATA (IEDT(114,J),J=1,7)/11, 1,15,
     &  6H$      ,6H BREAK,2H*6H       /
      DATA (IEDT(115,J),J=1,7)/ 3, 0,26,
     &  2H*6H       ,2H*6H      /
      DATA (IEDT(116,J),J=1,7)/ 3, 0, 5,
     &  6H***** ,3*6H      /
      DATA (IEDT(117,J),J=1,7)/ 3, 0,15,
     &  2H*6H       ,2H*6H      /
      DATA (IEDT(118,J),J=1,7)/ 5, 0, 1,
     &  6HV      ,3*6H      /
      DATA (IEDT(119,J),J=1,7)/ 5, 0, 1,
     &  6HW      ,3*6H      /
      DATA (IEDT(120,J),J=1,7)/ 3, 0, 4,
     &  6H0000   ,3*6H      /
      DATA (IEDT(121,J),J=1,7)/ 5, 0, 1,
     &  6HY      ,3*6H      /
```

```
251          DATA (IEDT(122,J),J=1,7)/ 5, 0, 1,
252       &  6HZ      ,3*6H      /
253          DATA (IEDT(123,J),J=1,7)/ 5, 0, 1,
254       &  6H       ,3*6H      /
255          DATA (IEDT(124,J),J=1,7)/ 3, 0,20,
256       &  6H******,6H******,6H******,6H**     /
257          DATA (IEDT(125,J),J=1,7)/ 5, 0, 1,
258       &  6H%      ,3*6H      /
259          DATA (IEDT(126,J),J=1,7)/ 5, 0, 1,
260       &  6H=      ,3*6H      /
261          DATA (IEDT(127,J),J=1,7)/ 5, 0, 1,
262       &  6H"      ,3*6H      /
263          DATA (IEDT(128,J),J=1,7)/ 5, 0, 1,
264       &  6H½      ,3*6H      /
265C
266          DATA IEOR/ 31/
267          DATA IREP/ 68/
268          DATA ITABLE/  0, 63,  0/
269C
270C        PROGRAM VARIABLES
271C
272          DIMENSION IDATE(2),INF(9),IDF(9),INPUT(80),
273       &   IOUT(80),ISS(2)
274C
275          CHARACTER CNCDFN*12,CDCDFN*12
276C
277          EQUIVALENCE (CNCDFN,INF(5)),(CDCDFN,IDF(5))
278C
279          DATA IBLANK/16/
280          DATA IGOST/O400000000000/
281          DATA IMOD/0/
282          DATA IPRMRD/1/
283          DATA IPRMWR/3/
284          DATA (INF(I),I=1,3)/6HZZZSBO,6HDFILES,6H    /  /
285          DATA (INF(I),I=7,9)/2*6H       ,6H"20" ;/
286          DATA (IDF(I),I=1,3)/6HZZZSBO,6HDFILES,6H    /  /
287          DATA (IDF(I),I=7,9)/2*6H       ,6H"30" ;/
288          DATA ISS/O123124101124,O073040040040/
289          DATA I20/20/
290          DATA I30/30/
291C
292       1  FORMAT(1X,A6,A2,2X,F6.3)
293       2  FORMAT(V)
294       3  FORMAT(80R1)
295       4  FORMAT(1X,"INPUT = ",I6,"OUTPUT = ",I6)
296       5  FORMAT(14A6)
297       6  FORMAT(1X,14A6)
298       7  FORMAT(2X,4(O12,3X)/2X,4(O12,3X))
299C
300          CALL CALSUB(ISS)
```

```
301         CALL FPARAM(1,80)
302         CALL FMEDIA(20,0)
303C
304C        HEAD
305C
306         PRINT,"START DECODE PROGRAM"
307         CALL DATIM(IDATE,TIME)
308         WRITE(6,1) IDATE,TIME
309         WRITE(6,1)
310C
311C         READ ENCODE AND DECODE FILENAMES
312C
313         PRINT," "
314         PRINT,"ENTER ENCODE, DECODE FILENAMES"
315         READ(5,2) CNCDFN,CDCDFN
316C
317C         ATTACH ENCODED FILE
318C
319         CALL ATTACH(I20,INF,IPRMRD,IMOD,ISTAT,)
320         IF((ISTAT.EQ.0).OR.(ISTAT.EQ.IGOST)) GO TO 100
321         PRINT,"COULD NOT ATTACH ENCODED FILE"
322         GO TO 999
323C
324C         ATTACH DECODE FILE
325C
326   100   CALL ATTACH(I30,IDF,IPRMWR,IMOD,ISTAT,)
327         IF((ISTAT.EQ.0).OR.(ISTAT.EQ.IGOST)) GO TO 120
328         PRINT,"COULD NOT ATTACH DECODE FILE"
329         GO TO 999
330C
331C         READ ENCODED RECORD
332C
333   120   DO 125 IDUM=1,80
334   125   IOUT(IDUM) = IBLANK
335         READ(20,3,END=300) INPUT
336         IREC = IREC + 1
337C
338C         CHECK FOR END OF RECORD, (EOR)
339C
340         IF(INPUT(1).NE.IEOR) GO TO 130
341         INCT = INCT + 1
342         GO TO 120
343   130   IF(IEOR.LT.64) GO TO 150
344         DO 140 JCT=2,IEEDT/64
345         IF(INPUT(1).NE.ITABLE(JCT)) GO TO 140
346         ITABPO = JCT
347         ICHAR = 64*(ITABPO-1)+INPUT(2)
348         IF(ICHAR.NE.IEOR) GO TO 150
349         INCT = INCT + 2
350         GO TO 120
```

```
140   CONTINUE
C
C     DETERMINE END OF RECORD
C
150   DO 153 ICT=1,80
      IF(INPUT(81-ICT).EQ.63) GO TO 156
153   CONTINUE
156   DO 158 ICT1=1,6
      IF(INPUT(81-ICT-ICT1).NE.IBLANK) GO TO 159
158   CONTINUE
159   ILAST = 81-ICT-ICT1
      INCT = INCT + ILAST
C
C     PROCESS ENCODED RECORD
C
C     SELECT PROPER ENCODING TABLE
C
      ILOOP = 1
      IOUTPO = 1
160   ITABPO = 1
      DO 170 JCT=2,IEEDT/64
      IF(INPUT(ILOOP).NE.ITABLE(JCT)) GO TO 170
      ITABPO = JCT
      ILOOP = ILOOP + 1
      GO TO 180
170   CONTINUE
180   ICHAR = 64*(ITABPO-1) + INPUT(ILOOP) + 1
      IF(ICHAR.NE.IREP+1) GO TO 220
C
C     PROCESS REPEATED STRINGS NOT REPRESENTED
C             BY A SPECIAL SYMBOL
C
      IF(INPUT(ILOOP+1).NE.0) GO TO 190
      ICREP = 64 + INPUT(ILOOP+2)
      ILOOP = ILOOP + 3
      GO TO 200
190   ICREP = INPUT(ILOOP+1)
      ILOOP = ILOOP + 2
200   DO 210 ICT=1,ICREP
      IOUT(IOUTPO+ICT-1) = INPUT(ILOOP)
210   CONTINUE
      ILOOP = ILOOP + 1
      IOUTPO = IOUTPO + ICREP
      IF(ILOOP-ILAST) 160,160,290
C
C     PROCESS ALL OTHER STRINGS AND SYMBOLS
C
220   IF(IEDT(ICHAR,1).GT.8) IEDT(ICHAR,1) = 4
      GO TO (230,230,240,260,280,230,230,230),
230   PRINT,"TRANSFER ERROR"
```

```
401         PRINT,"(REC. NO., CHAR. POSITION, CHARACTER)"
402         WRITE(6,2) IREC
403         WRITE(6,2) ILOOP
404         WRITE(6,2) ICHAR
405         GO TO 999
406C
407C         PROCESS REPEATED STRINGS REPRESENTED BY
408C                   SPECIAL SYMBOLS
409C
410   240   IDCHAR = FLD(0,6,IEDT(ICHAR,4))
411         DO 250 ICT=1,IEDT(ICHAR,3)
412         IOUT(IOUTPO+ICT-1) = IDCHAR
413   250   CONTINUE
414         ILOOP = ILOOP + 1
415         IOUTPO = IOUTPO + IEDT(ICHAR,3)
416         IF(ILOOP-ILAST) 160,160,290
417C
418C         PROCESS OTHER STRINGS
419C
420   260   DO 270 ICT=1,IEDT(ICHAR,3)
421         IWD = 1 + (ICT-1)/6
422         INUM = 6*(ICT-1)
423         IREF = INUM/36
424         ISTART = INUM - 36*IREF
425         IOUT(IOUTPO+ICT-1) = FLD(ISTART,6,
426        &    IEDT(ICHAR,3+IWD))
427   270   CONTINUE
428         ILOOP = ILOOP + 1
429         IOUTPO = IOUTPO + IEDT(ICHAR,3)
430         IF(ILOOP-ILAST) 160,160,290
431C
432C         PROCESS ALPHABET
433C
434   280   IOUT(IOUTPO) = FLD(0,6,IEDT(ICHAR,4))
435         ILOOP = ILOOP + 1
436         IOUTPO = IOUTPO + 1
437        IF(ILOOP-ILAST) 160,160,290
438C
439C         WRITE OUTPUT TO FILE
440C
441   290   WRITE(30,3) (IOUT(I),I=1,79)
442         IOUTTO = IOUTTO + 80
443         GO TO 120
444C
445C         OUTPUT SUMMARY STATISTICS
446C
447   300   WRITE(6,4) INCT,IOUTTO
448C
449C         TAIL
450C
```

```
451  999    PRINT,"STOP DECODE PROGRAM"
452         CALL DATIM(IDATE,TIME)
453         WRITE(6,1) IDATE,TIME
454         CALL CALSUB(ISS)
455         STOP
456         END
```

APPENDIX C. EXECUTION RESULTS

JCL RAP EXECUTION RESULTS

SOURCE LANGUAGE SYNTAX TABLE

```
   1      1 15  $        #######                    0
   1      1 15  %        #######                    0
   2      0  4  NNSP                                0
   2      0  4  NNST                                0
   2      0  5  DEN16                               0
   2      0 18  **,R,R,ZZZOBJ-NNS/                  0
```

INPUT FILE STATISTICS

CONTROL CARDS	5311
COMMENT/DATA CARDS	1293
TOTAL CARDS	6604

0 =	6803	1 =	5549	2 =	3054	3 =	1792
4 =	1458	5 =	1484	6 =	1778	7 =	782
8 =	659	9 =	1329	=	0	# =	27
@ =	2	: =	20	=	0	? =	2
=	69982	A =	4632	B =	2169	C =	3083
D =	5395	E =	6871	F =	2922	G =	565
H =	550	I =	4830	& =	19	. =	600
=	1	(=	7	=	2	- =	1
=	1	J =	1203	K =	1202	L =	4487
M =	2473	N =	7507	O =	4099	P =	3581
Q =	56	R =	5021	- =	1231	$ =	5387
* =	9771	) =	7	; =	9	' =	1
+ =	22	/ =	2059	S =	6520	T =	8909
U =	1754	V =	481	W =	523	X =	1467
Y =	404	Z =	1620	=	1	, =	12208
% =	486	= =	455	¬ =	207	=	6605

SUM = 216125 EOR CHAR. = 6604 BLANKS SUPP. = 312195

HMAX = 6.0000 HC0 = 4.2510 HC1 = 2.4697

INTERMEDIATE STATISTICS

NUMBER OF TABLES = 2

SYMBOLS AND STRINGS BEFORE ASSIGNMENT

TP	WGH	CH	STRING		FREQUENCY
4	16044	15	$	FILE	1146
4	10766	15	$	TAPE9	769
4	7252	15	$	LIMITS	518
5	6547	1	,		6547
5	5588	1	T		5588
4	5194	15	$	IDENT	371
4	5138	15	$	PRMFL	367
5	4750	1	1		4750
4	4634	15	$	SELECT	331
5	4199	1	S		4199
5	4175	1	0		4175
5	4104	1	R		4104
4	4102	15	$	EXECUTE	293
5	3989	1	D		3989
5	3370	1	A		3370
4	3080	15	$	IF	220
4	3052	5	DEN16		763
5	3029	1	O		3029
5	3023	1	2		3023
5	2711	1			2711
5	2420	1	C		2420
4	2086	15	$	PROGRAM	149
4	2019	4	NNST		673
5	2012	1	/		2012
5	1944	1	B		1944
5	1883	1	L		1883
4	1862	15	%	INFO	133
5	1788	1	3		1788
5	1752	1	N		1752
5	1748	1	E		1748
4	1659	4	NNSP		553
5	1655	1	I		1655
5	1583	1	P		1583
3	1558	2	,,		1558
5	1488	1	5		1488
5	1482	1	4		1482
5	1296	1	M		1296
6	1279	0	EOR		1279
4	1274	15	$	LOWLOAD	91

```
5   1248   1  U                                1248
4   1204  15  $       ENDJOB                    86
5   1196   1  K                                1196
4   1190  15  %       JOBNAME                   85
5   1176   1  X                                1176
4   1148  15  $       USE                       82
3   1144   2  00                               1144
3   1132   4  ,,,,                              566
4   1078  15  $       READ                      77
4   1078  15  $       OPTION                    77
5   1076   1  *                                1076
5   1056   1  6                                1056
5    997   1  -                                 997
5    989   1  J                                 989
5    980   1  F                                 980
4    966  15  $       WHEN                      69
5    820   1  7                                 820
4    799  18  **,R,R,ZZZOBJ-NNS/                 47
4    798  15  %       WHEN                      57
4    770  15  %       IF                        55
3    714   2  NN                               714
4    714  15  $       INPUT                     51
5    680   1  8                                680
3    614   3  ZZZ                              307
4    602  15  $       CONVER                    43
5    593   1  .                                593
4    588  15  $       SYSOUT                    42
4    588  15  $       NOTE                      42
5    567   1  9                                567
4    560  15  $       DATA                      40
5    551   1  Z                                551
7    485   0  REP                              485
5    455   1  =                                455
5    447   1  V                                447
5    429   1  H                                429
4    406  15  $       MSG2                      29
4    406  15  $       GOTO                      29
5    349   1  $                                349
5    347   1  G                                347
3    346  27                                   173
4    336  15  $       LIMIT                     24
5    335   1  Y                                335
5    307   1  W                                307
3    304   6                                   152
4    280  15  %       GOTO                      20
3    272   4                                   136
4    252  15  $       FUTIL                     18
3    220   5                                   110
3    218   3                                   109
5    216   1  "                                216
```

```
3    182   3   ***                                91
4    182  15   $       191PK                      13
3    172  29                                      86
3    172   3   000                                86
4    168  15   $       LIBRARY                    12
4    154  15   $       PUNCH                      11
5    146   1   %                                 146
4    126  15   $       UTILITY                     9
4    126  15   $       FORM                        9
4    112  15   $       TAPE7                       8
3    100  28                                      50
3     93  67   *************************          31
4     84  15   $       SET                         6
4     84  15   $       BREAK                       6
3     76  26                                      38
3     74   5   *****                              37
3     68  15                                      34
3     62   4   0000                               31
3     58  20   ********************               29
5     56   1   Q                                  56
5     54   1                                      54
5     37   1   :                                  37
5     36   1   #                                  36
5     34   1   @                                  34
5     22   1   +                                  22
5     21   1                                      21
5     20   1   &                                  20
5     16   1                                      16
5     13   1   -                                  13
5     10   1   (                                  10
5      9   1   ;                                   9
5      7   1   )                                   7
5      6   1                                       6
5      5   1   ?                                   5
5      1   1                                       1
5      1   1                                       1
5      1   1   _                                   1
5      1   1   '                                   1
```

ENCODING/DECODING TABLES

TABLE 1

0	5	4175	1	0		4175
1	5	4750	1	1		4750
2	5	3023	1	2		3023
3	5	1788	1	3		1788
4	5	1482	1	4		1482
5	5	1488	1	5		1488
6	5	1056	1	6		1056
7	5	820	1	7		820
8	5	680	1	8		680
9	4	16044	15	$	FILE	1146
	4	10766	15	$	TAPE9	769
#	4	7252	15	$	LIMITS	518
@	4	5194	15	$	IDENT	371
:	4	5138	15	$	PRMFL	367
	4	4634	15	$	SELECT	331
?	4	4102	15	$	EXECUTE	293
	5	2711	1			2711
A	5	3370	1	A		3370
B	5	1944	1	B		1944
C	5	2420	1	C		2420
D	5	3989	1	D		3989
E	5	1748	1	E		1748
F	5	980	1	F		980
G	4	3080	15	$	IF	220
H	4	3052	5	DEN16		763
I	5	1655	1	I		1655
&	4	2086	15	$	PROGRAM	149
.	4	2019	4	NNST		673
	4	1862	15	%	INFO	133
(	4	1659	4	NNSP		553
	3	1558	2	,,		1558
-	6	1279	0	EOR		1279
	4	1274	15	$	LOWLOAD	91
J	5	989	1	J		989
K	5	1196	1	K		1196
L	5	1883	1	L		1883
M	5	1296	1	M		1296
N	5	1752	1	N		1752
O	5	3029	1	O		3029
P	5	1583	1	P		1583
Q	4	1204	15	$	ENDJOB	86
R	5	4104	1	R		4104
-	5	997	1	-		997
$	4	1190	15	%	JOBNAME	85

*	5	1076	1	*		1076
)	4	1148	15	$	USE	82
;	3	1144	2	00		1144
'	3	1132	4	,,,,		566
+	4	1078	15	$	READ	77
/	5	2012	1	/		2012
S	5	4199	1	S		4199
T	5	5588	1	T		5588
U	5	1248	1	U		1248
V	4	1078	15	$	OPTION	77
W	4	966	15	$	WHEN	69
X	5	1176	1	X		1176
Y	4	799	18	**,R,R,ZZZOBJ-NNS/		47
Z	4	798	15	%	WHEN	57
	4	770	15	%	IF	55
¬	5	6547	1	,		6547
%	3	714	2	NN		714
=	4	714	15	$	INPUT	51
"	3	614	3	ZZZ		307
	8	0	0	TABLE SHIFT 2		0

TABLE 2

0	4	602	15	$	CONVER	43
1	4	588	15	$	SYSOUT	42
2	4	588	15	$	NOTE	42
3	4	560	15	$	DATA	40
4	7	485	0	REP		485
5	4	406	15	$	MSG2	29
6	4	406	15	$	GOTO	29
7	3	346	27			173
8	4	336	15	$	LIMIT	24
9	5	567	1	9		567
	5	54	1			54
#	5	36	1	#		36
@	5	34	1	@		34
:	5	37	1	:		37
	5	21	1			21
?	5	5	1	?		5
	3	304	6			152
A	4	280	15	%	GOTO	20
B	3	272	4			136
C	4	252	15	$	FUTIL	18
D	3	220	5			110
E	3	218	3			109
F	3	182	3	***		91
G	5	347	1	G		347
H	5	429	1	H		429
I	4	182	15	$	191PK	13

&	5	20	1	&		20
.	5	593	1	.		593
	5	1	1			1
(	5	10	1	(		10
	5	16	1			16
-	5	13	1	-		13
	5	6	1			6
J	3	172	29			86
K	3	172	3	000		86
L	4	168	15	$	LIBRARY	12
M	4	154	15	$	PUNCH	11
N	4	126	15	$	UTILITY	9
O	4	126	15	$	FORM	9
P	4	112	15	$	TAPE7	8
Q	5	56	1	Q		56
R	3	100	28			50
-	3	93	67	**********		31
$	5	349	1	$		349
*	4	84	15	$	SET	6
)	5	7	1	)		7
;	5	9	1	;		9
'	5	1	1	'		1
+	5	22	1	+		22
/	4	84	15	$	BREAK	6
S	3	76	26			38
T	3	74	5	*****		37
U	3	68	15			34
V	5	447	1	V		447
W	5	307	1	W		307
X	3	62	4	0000		31
Y	5	335	1	Y		335
Z	5	551	1	Z		551
	5	1	1			1
¬	3	58	20	**********		29
%	5	146	1	%		146
=	5	455	1	=		455
"	5	216	1	"		216
	5	1	1			1

NUMBER OF ENTRIES IN TABLE 2 IS 7131

OUTPUT STATISTICS - BLANKS REMOVED & STRINGS SUBSTITUTED

0 =	4218	1 =	4792	2 =	3065	3 =	1828
4 =	1967	5 =	1517	6 =	1085	7 =	993
8 =	704	9 =	1713	=	823	# =	554
@ =	405	: =	404	=	352	? =	298
=	2863	A =	3390	B =	2080	C =	2438
D =	4099	E =	1857	F =	1071	G =	567
H =	1192	I =	1668	& =	169	. =	1266
=	134	(=	563	=	1574	- =	1292
=	97	J =	1075	K =	1282	L =	1895
M =	1307	N =	1761	O =	3038	P =	1591
Q =	142	R =	4154	- =	1028	$ =	434
* =	1082	) =	89	; =	1153	' =	567
+ =	99	/ =	2018	S =	4237	T =	5625
U =	1282	V =	524	W =	376	X =	1207
Y =	382	Z =	608	=	56	, =	6576
% =	860	= =	506	¬ =	523	=	7132

SUM = 103647 EOR CHAR. = 1279 BLANKS SUPP. = 312195

HC0 = 5.3978 BLANKS REMOVED - STRINGS SUBSTITUTED

AVERAGE OF 2.8774 BITS/SYMBOL

FORTRAN RAP EXECUTION RESULTS

SOURCE LANGUAGE SYNTAX TABLE

FILE	SEQUENCED			
2	0	5	CALL	0
2	0	10	CHARACTER	0
2	0	7	COMMON	0
2	0	8	CONTINUE	0
2	0	5	DATA	0
2	0	7	DECODE	0
2	0	7	DECODE(	0
2	0	10	DIMENSION	0
2	0	3	DO	0
2	0	17	DOUBLE PRECISION	0
2	0	4	END	0
2	0	12	EQUIVALENCE	0
2	0	7	FORMAT	0
2	0	9	FUNCTION	0
2	0	6	GO TO	0
2	0	6	PRINT	0
2	0	6	PRINT,	0
2	0	5	READ	0
2	0	5	READ(	0
2	0	5	READ,	0
2	0	7	RETURN	0
2	0	5	STOP	0
2	0	11	SUBROUTINE	0
2	0	6	WRITE	0
2	0	6	WRITE(	0
2	0	3	=	0
2	0	3	+	0
2	0	3	-	0
2	0	4	.EQ.	0
2	0	4	.NE.	0
2	0	3	=1,	0

INPUT FILE STATISTICS

0 =	1237	1 =	1587	2 =	688	3 =	488
4 =	483	5 =	455	6 =	1392	7 =	318
8 =	370	9 =	180	=	4	# =	20
@ =	2	: =	3	=	20	? =	2
=	19833	A =	1553	B =	234	C =	1787
D =	1260	E =	1545	F =	656	G =	403
H =	1038	I =	3080	& =	193	. =	393
=	4	(=	1649	=	9	- =	1
=	1	J =	555	K =	417	L =	810
M =	354	N =	1276	O =	1666	P =	614

Q =	86	R =	1096	- =	168	$ =	66
* =	130	) =	1646	; =	13	' =	2
+ =	238	/ =	679	S =	936	T =	2721
U =	564	V =	50	W =	147	X =	199
Y =	180	Z =	37	=	1	, =	3123
% =	12	= =	844	¬ =	213	=	1953

SUM = 61684 DEL CHAR. = 1947 EOR CHAR. = 4

HMAX = 6.0000 HC0 = 4.3316 HC1 = 2.6977

DEL CHAR. = 1955
SEQ. CHAR. REMOVED = 5417
SEQ. CHAR. LEFT = 4

INTERMEDIATE STATISTICS

NUMBER OF TABLES = 2

SYMBOLS AND STRINGS BEFORE ASSIGNMENT

TP	WGH	CH	STRING	FREQUENCY
5	2826	1	I	2826
5	2793	1	,	2793
5	2013	1	T	2013
6	1955	0	DEL	1955
5	1648	1	C	1648
5	1639	1	)	1639
5	1636	1		1636
5	1550	1	(	1550
3	1406	8		703
5	1392	1	6	1392
5	1273	1	1	1273
5	1137	1	E	1137
4	1104	5	DATA	276
5	1053	1	O	1053
5	1032	1	H	1032
5	971	1	N	971
3	942	3		471
4	940	6	GO TO	188
5	935	1	A	935
5	920	1	R	920
5	918	1	S	918
3	898	6		449
5	849	1	0	849
5	795	1	D	795
5	757	1	L	757
4	754	3	=	377
3	682	5		341
5	682	1	2	682
5	673	1	/	673
5	649	1	F	649
4	616	3	=1,	308
4	602	8	CONTINUE	86
5	574	1	P	574
5	555	1	J	555
5	488	1	3	488
5	477	1	4	477
3	474	7		237
5	473	1	U	473
5	455	1	5	455

```
3    434   9                                  217
4    420   6  WRITE(                           84
5    417   1  K                               417
5    370   1  8                               370
5    335   1  M                               335
5    318   1  7                               318
3    292   2                                  292
5    234   1  B                               234
4    228   3  DO                              114
4    219   4  .EQ.                             73
5    215   1  G                               215
5    213   1  "                               213
5    199   1  X                               199
4    196   3   +                               98
5    193   1  &                               193
3    190   4                                   95
5    180   1  Y                               180
5    159   1  =                               159
5    159   1  9                               159
5    149   1  .                               149
4    147   4  .NE.                             49
5    140   1  +                               140
5    128   1  -                               128
3    116   6  000000                           58
4    108  10  DIMENSION                        12
4    105   6  WRITE                            21
4     96   5  CALL                             24
4     85   6  PRINT,                           17
4     85   6  PRINT                            17
5     80   1  *                                80
4     80   3   -                               40
5     66   1  $                                66
4     63   4  END                              21
4     55  12  EQUIVALENCE                       5
4     54  10  CHARACTER                         6
5     45   1  V                                45
5     42   1  W                                42
4     42   7  DECODE                           7
4     42   7  FORMAT                           7
4     40   5  READ                            10
4     30   7  DECODE(                          5
4     28   5  READ(                            7
4     24   5  STOP                             6
5     20   1                                   20
3     16   3  ZZZ                              8
3     14   3  999                              7
3     14   6  ******                           7
5     13   1  ;                                13
5     13   1  Z                                13
3     12  11                                    6
```

```
5     12   1  %                                          12
5     10   1  #                                          10
5      9   1                                             9
5      8   1  Q                                          8
3      8  18                                             4
3      6  11  00000000000                                3
3      4   3  444                                        2
4      4   5  READ,                                      1
3      4   3  222                                        2
3      4   5  #####                                      2
3      4   3  ///                                        2
5      4   1                                             4
5      4   1                                             4
3      4   3  111                                        2
5      3   1  :                                          3
3      2   3  (((                                       1
5      2   1  '                                          2
3      2  15                                             1
3      2  16                                             1
3      2   3  000                                        1
5      2   1  @                                          2
5      2   1  ?                                          2
3      2   3  ***                                        1
3      2  19                                             1
3      2   5  *****                                      1
3      2   4  ,,,,                                       1
3      2  23                                             1
3      2  22                                             1
3      2   4  ))))                                       1
5      2   1                                             2
3      2   3  )))                                       1
3      2   4  0000                                       1
5      1   1                                             1
5      1   1  =                                          1
5      1   1                                             1
2      0   9  FUNCTION                                   0
7      0   0  REP                                        0
3      0  21                                             1
```

ENCODING/DECODING TABLES

TABLE 1

0	5	849	1	0	849
1	5	1273	1	1	1273
2	5	682	1	2	682
3	5	488	1	3	488
4	5	477	1	4	477
5	5	455	1	5	455
6	5	1392	1	6	1392
7	5	318	1	7	318
8	5	370	1	8	370
9	5	159	1	9	159
	6	1955	0	DEL	1955
#	3	1406	8		703
@	4	1104	5	DATA	276
:	3	942	3		471
	4	940	6	GO TO	188
?	3	898	6		449
	5	1636	1		1636
A	5	935	1	A	935
B	5	234	1	B	234
C	5	1648	1	C	1648
D	5	795	1	D	795
E	5	1137	1	E	1137
F	5	649	1	F	649
G	5	215	1	G	215
H	5	1032	1	H	1032
I	5	2826	1	I	2826
&	5	193	1	&	193
.	5	149	1	.	149
	4	754	3	=	377
(	5	1550	1	(	1550
	3	682	5		341
-	4	616	3	=1,	308
	4	602	8	CONTINUE	86
J	5	555	1	J	555
K	5	417	1	K	417
L	5	757	1	L	757
M	5	335	1	M	335
N	5	971	1	N	971
O	5	1053	1	O	1053
P	5	574	1	P	574
Q	3	474	7		237
R	5	920	1	R	920
-	5	128	1	-	128
$	3	434	9		217

*	4	420	6	WRITE(	84
)	5	1639	1	)	1639
;	3	292	2		292
'	4	228	3	DO	114
+	5	140	1	+	140
/	5	673	1	/	673
S	5	918	1	S	918
T	5	2013	1	T	2013
U	5	473	1	U	473
V	4	219	4	.EQ.	73
W	4	196	3	+	98
X	5	199	1	X	199
Y	5	180	1	Y	180
Z	3	190	4		95
	4	147	4	.NE.	49
_	5	2793	1	,	2793
%	3	116	6	000000	58
=	5	159	1	=	159
"	5	213	1	"	213
	8	0	0	TABLE SHIFT 2	0

TABLE 2

0	4	108	10	DIMENSION	12
1	4	105	6	WRITE	21
2	4	96	5	CALL	24
3	4	85	6	PRINT,	17
4	4	85	6	PRINT	17
5	4	80	3	-	40
6	4	63	4	END	21
7	4	55	12	EQUIVALENCE	5
8	4	54	10	CHARACTER	6
9	4	42	7	DECODE	7
	5	4	1		4
#	5	10	1	#	10
@	5	2	1	@	2
:	5	3	1	:	3
	5	20	1		20
?	5	2	1	?	2
	4	42	7	FORMAT	7
A	4	40	5	READ	10
B	4	30	7	DECODE(	5
C	4	28	5	READ(	7
D	4	24	5	STOP	6
E	3	16	3	ZZZ	8
F	3	14	3	999	7
G	3	14	6	******	7
H	3	12	11		6
I	3	8	18		4

&	3	6	11	00000000000	3
.	3	4	3	444	2
	5	4	1		4
(	4	4	5	READ,	1
	5	9	1		9
-	5	1	1	-	1
	5	1	1		1
J	3	4	3	222	2
K	3	4	5	#####	2
L	3	4	3	///	2
M	3	4	3	111	2
N	3	2	3	(((	1
O	3	2	15		1
P	3	2	16		1
Q	5	8	1	Q	8
R	3	2	3	000	1
-	3	2	3	***	1
$	5	66	1	$	66
*	5	80	1	*	80
)	3	2	19		1
;	5	13	1	;	13
'	5	2	1	'	2
+	3	2	5	*****	1
/	3	2	4	,,,,	1
S	3	2	23		1
T	3	2	22		1
U	3	2	4	))))	1
V	5	45	1	V	45
W	5	42	1	W	42
X	3	2	3	)))	1
Y	3	2	4	0000	1
Z	5	13	1	Z	13
	5	1	1		1
¬	2	0	9	FUNCTION	0
%	5	12	1	%	12
=	7	0	0	REP	0
"	3	0	21		1
	5	2	1		2

NUMBER OF ENTRIES IN TABLE 2 IS 605

OUTPUT STATISTICS - BLANKS REMOVED & STRINGS SUBSTITUTED

0 =	861	1 =	1294	2 =	706	3 =	505
4 =	494	5 =	495	6 =	1413	7 =	323
8 =	376	9 =	166	=	1959	# =	713
@ =	278	: =	474	=	208	? =	451
=	1643	A =	945	B =	239	C =	1655
D =	801	E =	1145	F =	656	G =	222
H =	1038	I =	2830	& =	196	. =	151
=	381	(=	1551	=	350	- =	309
=	87	J =	557	K =	419	L =	759
M =	337	N =	972	O =	1054	P =	575
Q =	245	R =	921	- =	129	$ =	283
* =	164	) =	1640	; =	305	' =	116
+ =	141	/ =	674	S =	919	T =	2014
U =	474	V =	118	W =	140	X =	200
Y =	181	Z =	108	=	50	, =	2793
% =	70	= =	159	¬ =	214	=	607

SUM = 42253 DEL CHAR. = 1955 EOR CHAR. = 0

HC0 = 5.4298 SEQ. CHAR. REMOVED - STRINGS SUBSTITUTED

AVERAGE OF 4.1099 BITS/SYMBOL

APPENDIX D. COMMON-CARRIER TARIFFS

American Telephone and Telegraph Tariffs

Cost of High-Density (HiD) Service at 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas (Currently Available)

	2.4 kbps	9.6 kbps
Houston Modem	55.00	200.00
Station Terminal	26.30	26.30
Channel Terminal	36.80	36.80
Mileage Charge (.89*1200)	1,068.00	1,068.00
Line-Conditioning (C-2)	0.00	40.00
Channel Terminal	36.80	36.80
Station Terminal	26.30	26.30
Newport News Modem	55.00	200.00
Total	$1,304.20	$1,634.20

Cost of Dataphone Digital Service at 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas (Unavailable at Present)

	2.4 kbps	9.6 kbps
Houston Data Service Unit	15.00	15.00
Digital Access Line	65.00	110.00
Mileage Charge (fixed)	20.00	60.00
+ (.4*1200 or .9*1200)	480.00	1,080.00
Digital Access Line	65.00	110.00
Newport News Data Service Unit	15.00	15.00
Total	$660.00	$1,390.00

Datran Tariffs

Cost of Facilities Capable of Transmitting Data at 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas via Atlanta, Georgia (currently Available)

Dataline 1	2.4 kbps	9.6 kbps
Houston Channel End Charge	81.00	139.50
Houston-Atlanta Channel (702 at .36 or .81)	252.72	568.62
Atlanta Channel End Charge	81.00	139.50
Atlanta Modem	55.00	250.00
Station & Channel Terminals	126.20	126.20
Mileage Charge (.89*510)	453.90	453.90
Line-Conditioning (C-2)	0.00	40.00
Newport News Modem	55.00	250.00
Total	$1,104.82	$1,967.72

Cost of Facilities Capable of Transmitting Data at 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas via Norfolk, Virginia (Unavailable at Present)

Dataline I	2.4 kbps	9.6 kbps
Houston Channel End Charge	81.00	139.50
Houston-Norfolk Channel (1206 miles at .36 or .81)	434.16	976.86
Newport News End Charge	103.50	157.50
Norfolk-Newport News (10 miles at .6 or 1.3)	6.00	13.00
Total	$624.66	$1,286.86

Datran Tariffs (Continued)

Dataline II	2.4 kbps	9.6 kbps
Houston Channel End Charge	81.00	139.50
Houston-Norfolk Channel (1206 miles at .54 or .81)	651.24	976.86
Channel Termination Charge	60.00	80.00
Local Distribution Charge (fixed)	50.00	50.00
(10 miles at $1.00)	10.00	10.00
Datalink Charge	40.00	55.00
Total	$892.24	$1,311.36

Cost of Facilities Capable of Transmitting Data at 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas via Washington, D. C. (currently available)

Dataline 1.	2.4 kbps	9.6 kbps
Houston Channel End Charge	81.00	139.50
Houston-Washington Channel (1217 miles at .36 or .81)	438.12	985.77
Washington Channel End Charge	81.00	139.50
Washington Modem	55.00	250.00
Station and Channel Terminals	126.20	126.20
Mileage Charge (.89*150)	133.50	133.50
Line-Conditioning (C-2)	0.00	40.00
Newport News Modem	55.00	250.00
Total	$969.82	$2,064.47

American Satellite Corporation Tariffs

Cost of Facilities Capable of Transmitting Data at 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas via Washington, D. C. (Currently Available)

	2.4 kbps	9.6 kbps
Houston Modem	55.00	250.00
Local Facility and Customer Termination Charge	45.00	45.00
Houston-Washington Channel	750.00	750.00
Channel Conditioning (C-2)	0.00	28.00
Local Facility and Customer Termination Charge	45.00	45.00
Station Terminal (Washington)	26.30	26.30
Channel Terminal	36.80	36.80
Mileage Charge (.89*150)	133.50	133.50
Line-Conditioning (C-2)	0.00	40.00
Channel Terminal	36.80	36.80
Station Terminal (Newport News)	26.30	26.30
Newport News Modem	55.00	250.00
Total	$1,209.70	$1,667.70

Telenet Tariffs

Cost of Facilities Capable of Transmitting Data at 2.4 and 9.6 kbps Between Newport News, Virginia and Houston, Texas via Washington, D. C. (Currently Available)

	2.4 kbps	9.6 kbps
Houston-Telenet Access Channel	290.00	700.00
Houston Access Port at TCO	200.00	200.00
Washington Access Port at TCO	200.00	200.00
Newport News Telenet	410.00	820.00
Sub-Total	$1,100.00	$1,920.00
Usage Charge (175 hours usage/ month, data sent at 50% effective rate using full packets)	354.00	1,418.00
Total	$1,454.00	$3,338.00

The vita has been removed from the scanned document

A HEURISTIC METHOD FOR REDUCING MESSAGE REDUNDANCY IN A FILE TRANSFER ENVIRONMENT

by

William Robert Bodwell

(ABSTRACT)

Intercomputer communications involves the transfer of information between intelligent hosts. Since communication costs are almost proportional to the amount of data transferred, the processing capability of the respective hosts might advantageously be applied through pre-processing and post-processing of data to reduce redundancy. The major emphasis of this research is development of the Substitution Method which minimizes data transfer between hosts required to reconstruct user JCL files, Fortran source files, and data files.

The technique requires that a set of user files for each category of files be examined to determine the frequency distribution of symbols, fixed strings, and repeated symbol strings to determine symbol and structural redundancy. Information gathered during the examination of these files when combined with the user created Source Language Syntax Table generate Encoding/Decoding Tables

which are used to reduce both symbol and structural redundancy. The Encoding/Decoding Tables allow frequently encountered strings to be represented by only one or two symbols through the utilization of table shift symbols. The table shift symbols allow less frequently encountered symbols of the original alphabet to be represented as an entry in a Secondary Encoding/Decoding Table. A technique is described which enables a programmer to easily modify his Fortran program such that he can take advantage of the Substitution Method's ability to compress data files by removing both informational and structural redundancy.

Each user file requested to be transferred is preprocessed at cost, C[prep], to reduce data (both symbol and structural redundancy) which need not be transferred for faithful reproduction of the file. The file is transferred over a noiseless channel at cost, C[ptran]. The channel consists of presently available or proposed services of the common-carriers and specialized common-carriers. The received file is post-processed to reconstruct the original source file at cost, C[post]. The costs associated with pre-processing, transferring, and post-processing are compared with the cost, C[otran], of transferring the entire file in its original form.