A Knowledge-Combined System Approach to Optimum Design of Type-I Hybrid ARQ/FEC Error Control Schemes

by

Qing Yang

B.S.E.E., University of Science and Technology of China (USTC), 1982

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in the Department of
Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

[redacted]

Dr. V. K. Bhargava, Supervisor

[redacted]

Dr. N. J. Dimopoulos, Departmental Member

[redacted]

Dr. H. A. Muller, Outside Member(Computer Science)

[redacted]

Dr. M. Serra, External Examiner(Computer Science)

ISBN 0-315-62375-6

Supervisor: Professor Vijay. K. Bhargava

Abstract

In this work, a systematic approach to optimum coding design for Type-I Hybrid ARQ/FEC error control schemes is developed. A knowledge-combined expert system method is proposed for optimal coding selection. Performance analysis is further discussed for Type-I Hybrid ARQ schemes. Particularly, a simple time delay analysis is carried out by using queueing theory. Finally, a pragmatic approach to design an optimum Type-I hybrid ARQ scheme with BCH error correcting code is developed by introducing overall throughput as a system performance measure. This optimization provides a further improvement of the overall throughput efficiency with reasonable implementation complexity which is found to have a local maximum in the throughput contour plot.

Examiners:

[redacted]

Dr. V. K. Bhargava, Supervisor

[redacted]

Dr. N. J. Dimopoulos, Departmental Member

[redacted]

Dr. H. A. Muller, Outside Member(Computer Science)

[redacted]

Dr. M. Serra, External Examiner(Computer Science)

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I am indebted to many people in the Department of Electrical and Computer Engineering for their help and encouragement throughout the development of this thesis. Particular acknowledgment must first go to my supervisor, Dr. Vijay K. Bhargava, who initially interested me in this subject. Without his advice and support, this work would never have been possible. I would also like to thank Dr. Wu-Sheng Lu for his helpful comments in the early stages of the thesis. A great deal of appreciation are given to my fellow members in Digital Communication Group, which is under guidance of Dr. Vijay K. Bhargava, for having a lot of interesting discussions in a creative atmosphere. The financial support from University of Victoria in the form of a fellowship is also greatly appreciated. Finally, special thanks are due to my wife, Qing Zhu, without whose patience, support and looking after my lovely baby daughter, this work would be much less completed.

Error control is a system design technique that can fundamentally change the trade-offs in a communications system design.

E. R. Berlekamp, *et al.*, 1987 [4].

In general, there is no systematic procedure for selecting codes for a particular application.

V. K. Bhargava, 1981 [7].

In trying to apply a computer to a task that humans do, we often discover that it doesn't work. One common problem is that humans are able to deal with fuzzy concepts, but computers are not. However, if we look closer at such tasks, we often discover that the weakness actually lies not in the computer but in ourselves—that we didn't understand what we were doing in the first place.

L. Earnest, 1989 [18].

What makes an expert system seem not truly intelligent? With today's technology, there are a number of limitations of expert systems that might make expert systems appear to some to be "artificially stupid". However, remember that expert systems are really just in their infancy of development.

J. Liebowitz, 1989 [29].

Like a science-fiction writer, I'm thinking, What if it were like this? or, Is there an interesting problem of this type?, and I'm not caring whether someone is working on it or not. It's usually just that I like to solve a problem, and I work on these all the time.

C. E. Shannon, 1984 [51].

Chapter 1

Introduction

1.1 Introduction

In digital communication, significant performance improvements may be obtained if error control coding is properly applied [6] [4] [32]. However, selection of a coding scheme for specific applications is often a complicated task [7] [44] [61]. The choice is affected by a set of system design goals. Some of these goals pose case-dependent conflicting requirements. In this work, a "knowledge-combined system approach" with "rule-transfer algorithms" is proposed and applied to optimal coding selection.

Performance analysis and numerical optimization are further added into the approach to achieve optimum coding design. Type-I hybrid ARQ/FEC error control techniques have been chosen as a case study to develop our approach.

1.2 Error Control for Digital Communications

The task of a digital communication designer is to provide a cost-effective system for transmitting information from a sender to a user at the rate and reliability that the user requires.

The key parameters of the design are transmission bandwidth, signal power, and complexity of the system implementation.

The information transmission rate and accuracy of the delivered information are typically determined by the user's requirements.

The transmission bandwidth is often constrained by factors specific to the particular transmission medium used. For example, telephone circuits are separated into nominal 3-kHz bandwidth segments due to longstanding engineering practice in the telephone industry. Similarly, there are standard bandwidths for individual channels on terrestrial radio circuits and satellite links due to established government regulations on spectrum utilization. But in some other cases such as links to and from vehicles in deep space, bandwidth constraints are not critical issues.

The signal power and implementation complexity are usually very much under the designer's control. In fact, possible trade-offs between power and complexity are central issues in the design task. Both characteristics represent cost factors for the designer to consider. For example, in most systems a desired level of accuracy in the delivered information can be achieved by supplying enough power in the transmitted signal to overcome channel disturbances that produce errors. An alternative to increasing signal power is to add systematic redundancy to the transmitted information in the form of error control coding. However, the use of coding adds complexity to the system, particularly for the implementation of the decoding operations. Since the addition of redundancy also implies the need to increase transmission bandwidth, the design trade-off must include considerations of bandwidth.

Two fundamental techniques for error control are used in digital communication systems: forward-error-control (FEC) schemes and automatic repeat request (ARQ) schemes [4] [32]. In ARQ techniques, an error-detecting code is used to detect whether a received vector contains errors. If so, a retransmission request is sent to the transmitter through a feedback channel. This procedure is continued until a codeword is received. In FEC

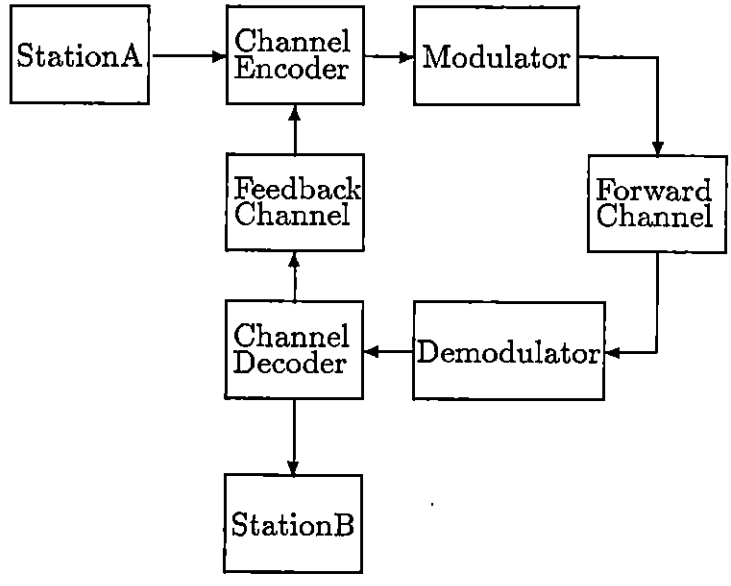


Figure 1.1: Basic Blocks of Hybrid ARQ/FEC Schemes for Digital Communications

techniques, a code is used for the correction of the error patterns which are frequently introduced by the transmission channel.

In data communications systems, such as packet-switching data networks or computer communications networks, ARQ techniques are often preferred to FEC techniques, because they present a simpler implementation and a higher performance, particularly for mean and high signal/noise ratios. In communications (or data storage) systems where feedback channels are not available or retransmission is not suitable for some reason, FEC is of course the only choice.

The application of ARQ to data communications is extensive, but the greatest disadvantage is that the throughput efficiency of ARQ schemes is

dependent on channel conditions. In particular, for low signal/noise ratios, the retransmission number required for correctly receiving a codeword may be high, and hence the transmission rate may be reduced in an unacceptable way. A good approach in such an application is to use a hybrid ARQ approach where some FEC coding is used to correct the more common errors and an ARQ scheme is used to drive the error rate down to the desired level. Basic blocks of hybrid ARQ/FEC schemes are shown in Figure 1.1. In such a hybrid approach, powerful random-error-correcting codes that take advantage of soft receiver decisions can be used; at the same time, the hard decision FEC decoded data can be used with standard ARQ protocols.

1.3 Features of Error Control Coding Design

According to the experience of experts in the communication industry, many communication engineers have difficulty in grasping the concepts of error control coding and how to apply coding in a cost-effective design. Error control coding does not lend itself readily to an intuitive understanding [38]. System designers are further perplexed by the plethora of classes and types of codes, with little or no insight provided as to which codes to use on which channels and how to select and tailor code parameters to a system at hand. The designers need to know which coding techniques are likely to be the most useful as well as how to select a coding technique to satisfy a particular system requirement.

Similar to scheme selection problems in many engineering system design processes, error control coding design for communication systems often face a situation where many different schemes are available to the system designer. This kind of scheme selection problems has the following common features:

- Each scheme offers its own system trade-offs;
- Some design goals pose case-dependent conflicting requirements;
- Information about the application is usually incomplete or uncertain;
- It is impossible to test all the available schemes to make a choice;

Therefore, scheme selection is a very complicated task which requires expert knowledge. As a result, one seeks to develop an expert system approach to solve the problem. Clearly, an expert system does not replace a human expert; however, it may play the role of a colleague or an assistant to the human.

1.4 Outline of Monograph

An outline of the remainder of this monograph is as follows. Chapter 2 describes the problem of error control coding selection for digital communications. The major factors affecting coding selection are summarized. Then a knowledge-combined expert system approach is defined in Chapter 3. In Chapter 4, we discuss two important issues for constructing an expert system using our novel rule-transfer algorithms. Systematic performance analysis is shown in Chapter 5 for a particular hybrid ARQ error control scheme. A pragmatic method to numerical optimization for a Type-I hybrid ARQ with BCH codes is then carried out in Chapter 6. Finally, Chapter 7 draws some conclusions about the work done and suggests some areas for future work.

Chapter 2

Description of the Coding Design Problem

2.1 Factors Affecting the Selection of Error Control Coding

In digital communication systems, two primary concerns are system reliability and efficiency. Often a less costly approach to improving the communication performance is to add error control coding at the transmit and receive terminals. But many different error control coding schemes are available to the designer of a digital communication system required for reliable data transmission. They are FEC schemes, ARQ schemes and an appropriate combination of FEC with ARQ(hybrid). Each scheme offers system trade-offs of its own. The final choice made by the designer depends on the following design goals:

- maximum data rate;
- minimum bit error rate;
- minimum transmitted power;
- maximum resistance to interference;
- minimum implementation complexity.

Some of these design goals are case-dependent conflicting requirements and the information about a particular application may be incomplete or uncertain so that the problem cannot be expressed in a fully formalized way.

The coding selection problem is first described in [7] for satellite communications, and later in [44] and [61]. Before attempting to select error control schemes in a particular application, we must consider the coding performance, the channel environment, and any implementation or system issues that could affect the choice. At least the following factors should be taken into consideration in coding selection problems:

1. Decoded bit error rate, i.e., the allowed rate of accepting incorrect messages.
2. Data rate required to transmit messages.
3. Available ratio of energy per bit to noise power density, i.e., E_b/N_0.
4. Communication channel characteristics, i.e., random errors, burst error or a combination of both.
5. Channel bit error rate.
6. Decoding speed (buffer at receiver may be required).
7. Availability and characteristics of a feedback channel.
8. Required information throughput rate.
9. Acceptable bandwidth expansion.
10. Required decoding performance, i.e., required coding gain.
11. Nature of message format.
12. Coding efficiency, i.e., code rate.

13. Channel round trip time delay.
14. Allowed throughput delay for channel encoding and decoding.
15. Characteristics of any mutual-user or hostile jamming.
16. Robustness of the coding scheme to channel variations that is required.
17. Availability of soft receiver decisions (quality information) for the decoder.
18. Decoder synchronization requirements.
19. Multidestination decoding capability.
20. Decoder cost and complexity.

2.2 General Coding Selection Process

The general *coding selection process* can be formulated as outlined in Figure 2.1, where user requirements are expressed by the design factors summarized above. The values of the factors may be special constants or functions in a particular application. Two coding selection steps in Figure 2.1 have different meanings for each scheme. For ARQ, they mean to choose candidate variations of transmission protocols. For FEC, they mean to choose candidate block or convolutional codes. For hybrid FEC/ARQ, they mean a combination of both.

2.3 Difficulties in Designing Error Control Coding

We have mentioned that selection of a coding scheme is often a complicated task. The choice is affected by a set of system design goals. From

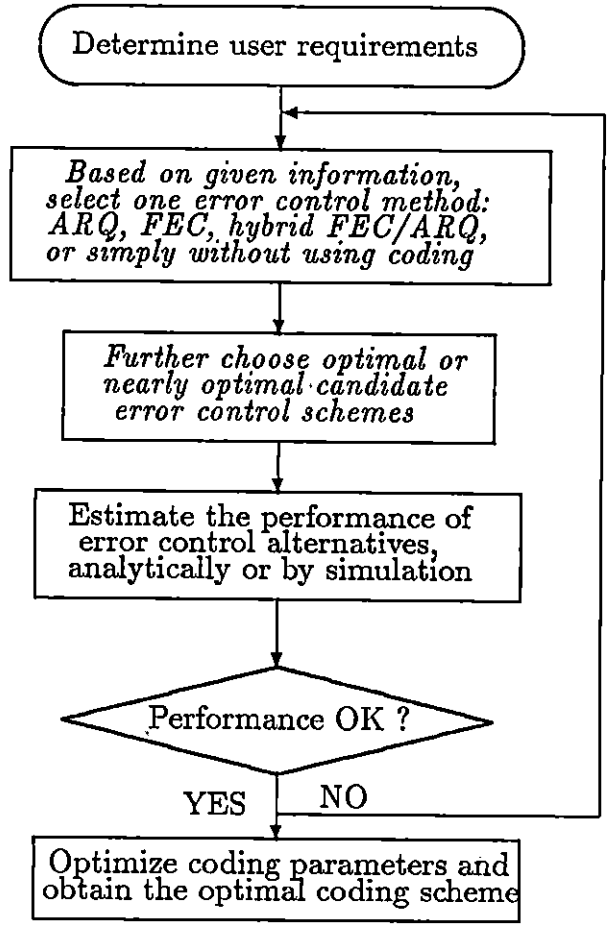


Figure 2.1: Description of Coding Selection Process

Figure 2.1 we can further see the following difficulties existing in coding design for a particular application:

- User requirements expressed by design factors may often be approximate descriptions with uncertain or incomplete information.
- Two coding selection steps are normally carried out by one or more coding experts based on their experience. No systematic method is available [7].
- In order to seek optimum coding selection, sometimes trial and error may be required.

All these characteristics make the task more difficult for communication system designers. Therefore, an expert system approach is expected to be able to assist humans dealing with the coding selection problem.

2.4 Summary

- Design of Error control coding for digital communications is affected by at least 20 factors which include user requirements, related performance issues and system parameters.
- There is no systematic method available for selecting optimum coding schemes for a particular application.

Chapter 3

A Knowledge-Combined System Approach

3.1 On Rule-Based Expert Systems

Rule-based deduction is essentially the "standard" inference model in expert systems. In this approach, rules always express a conditional, with an antecedent and a consequent component. The interpretation of a rule is that if the antecedent can be satisfied the consequent will be too. That is, domain-specific problem-solving knowledge is represented in rules which are basically of the form:

$$"IF < antecedents > THEN < consequents >"$$

When the consequent defines an action, the effect of satisfying the antecedent is to schedule the action for execution. When the consequent defines a conclusion, the effect is to infer the conclusion.

Although many different techniques have emerged for organizing collections of rules into automated experts, all rule-based system architectures share certain key properties as follows [60]:

- They incorporate practical human knowledge in conditional if-then rules.

- Their skill increases at a rate proportional to the enlargement of their knowledge bases.
- They can solve a wide range of possibly complex problems by selecting relevant rules and then combining the results in appropriate ways.
- They explain their conclusions by retracing their actual lines of reasoning and translating the logic of each rule employed into natural language.

The inference mechanism consists of a rule interpreter which, when given a specific set of problem features, determines applicable rules and applies them in some specified order to reach conclusions about the case at hand. Rule-based deduction can be performed in a variety of ways, and rules can be chained together to make multiple-step deductions. In addition, in many systems one can attach "certainty factors" to rules to capture probabilistic information, and a variety of mechanisms can be used to propagate certainty measures during problem solving.

There are two important ways in which rules can be used in a rule-based system: one is called *forward-chaining* and the other *backward-chaining* [60].

- Backward-chaining System—a conclusion is assumed and the inference engine works backward in an attempt to find the facts supporting that conclusion. As soon as it finds a valid conclusion, it moves to a subgoal for that conclusion and then tries to prove this. This type of search is often called a goal-driven or consequence-driven search. Backward chaining is used when the goals are known and are relatively few in number.
- Forward-chaining System—the inference engine starts with the facts, and then works forward to find a conclusion that is supported by the facts. As each new conclusion is reached, it is added to the working

memory. Forward search strategies are often called data-driven or antecedent-driven searches.

Rule-based model architectures constitute the best currently available means for codifying the problem-solving know-how of human experts. The use of rules allows for non-numeric, judgemental knowledge because one does not need exact probabilities. But rule-based deduction has the following limitations:

- It lacks a precise analytic foundation for deciding which problems are solvable.
- It is often very difficult to test the consistency of a rule set.
- If the system encounters a situation which is not covered by the set of rules, it does not know what conclusion to draw.
- Debugging and maintenance of a knowledge base is difficult and becomes harder with the growth of the number of rules.
- There is no simple method in helping to refine and correct the rule base. For example, an expert system had its number of rules increased from 100 to 400 just to get a 10 percent increase in performance [60].

3.2 Knowledge Representation by Rules

Despite the existence of limitations in rule-based deduction, the *rule* has been widely accepted as the most well-known type of knowledge representation technique. Rules provide a formal way of representing recommendations, directives, or strategies. They are often appropriate when the domain knowledge results from empirical associations developed through years of experience in solving problems in an area.

In our coding selection context, the difference between an inference rule and a practical observation can be illustrated by the following examples:

Example 1 (a practical observation) [4]: Based on the experience in similar applications, coding experts select the $(2,1)$ $K = 7$ convolutional code with Viterbi decoding algorithm as the best choice for the particular case when observed factors are given as follows:

1. The channel noise is *white* and *Gaussian.*
2. The demodulator provides reliable soft decision information.
3. The output user data only needs a low level of integrity such as an output BER of 10^{-3} to 10^{-7}.
4. The transmission speed is low enough to allow the decoder to perform a relatively large number of operations per bit.

Example 2 (inference rules) [44]: Sequential decoders are somewhat more sensitive to channel variations than Viterbi decoders, and they have larger throughput delays (due to required buffering). However, sequential decoding systems are capable of achieving very large E_b/N_0 coding gains.

We can easily rewrite both above examples having the same format as:

IF a set of factors, denoted by F_i $(i = 1, 2, ..., q)$, *are observed, THEN the top choice of coding alternatives are those schemes denoted by* D_j $(j = 1, 2, ..., p)$.

We realize that the only difference between a real observation and an inference rule is whether it contains noisy factors or not. A real expert coding selection observation often contains a few noisy factors which do not really determine the decision of the output coding. Instead, an inference rule contains only accurate relationship between input and output. Later in our approach, we use practical observations as training samples to refine expert system performance, and use rules in system settings for rapid prototyping as well as for knowledge refinement.

3.3 Simulation vs. Knowledge-Based Inference

As we mentioned in the previous section, today's expert systems largely use the rule-based system model. However, even though rule-based model is the perfect choice for building expert systems, rules only are not capable of capturing the full range of expert knowledge for engineering system design. The coding selection problem and many other engineering problems contain well-known mathematical models, which should be regarded as important pieces of knowledge about such a system. In the error control coding context, it is clear that people do not become experts only by way of acquiring more and more rules. Experts draw conclusions not only based on the basis of rules gained from the experience, but also from precedent and analogy, as well as numerical computation or simulation.

On the other hand, traditional engineering problem-solving programs rely heavily on simulation. In expert systems we use so-called "inference engines". Given a knowledge base for a particular domain, an expert system can draw conclusions about this domain. The knowledge base is composed of discrete (qualitative) knowledge. This differs from simulation programs, where the knowledge about a particular problem is represented in a quantitative model.

Even in situations where the full mathematical model is readily available, the expert system approach can still be beneficial, mainly due to the speed of drawing conclusions. Sometimes we do not really need to conduct all calculations; a simple rule may well do the job. It is always impractical to carry out all the simulations of existing coding schemes to decide the optimal coding scheme. Thus, we conclude that both qualitative and quantitative models are in some cases unnecessary, and in some other cases indispensable, but in a software system for engineering problem solving, both of them must coexist.

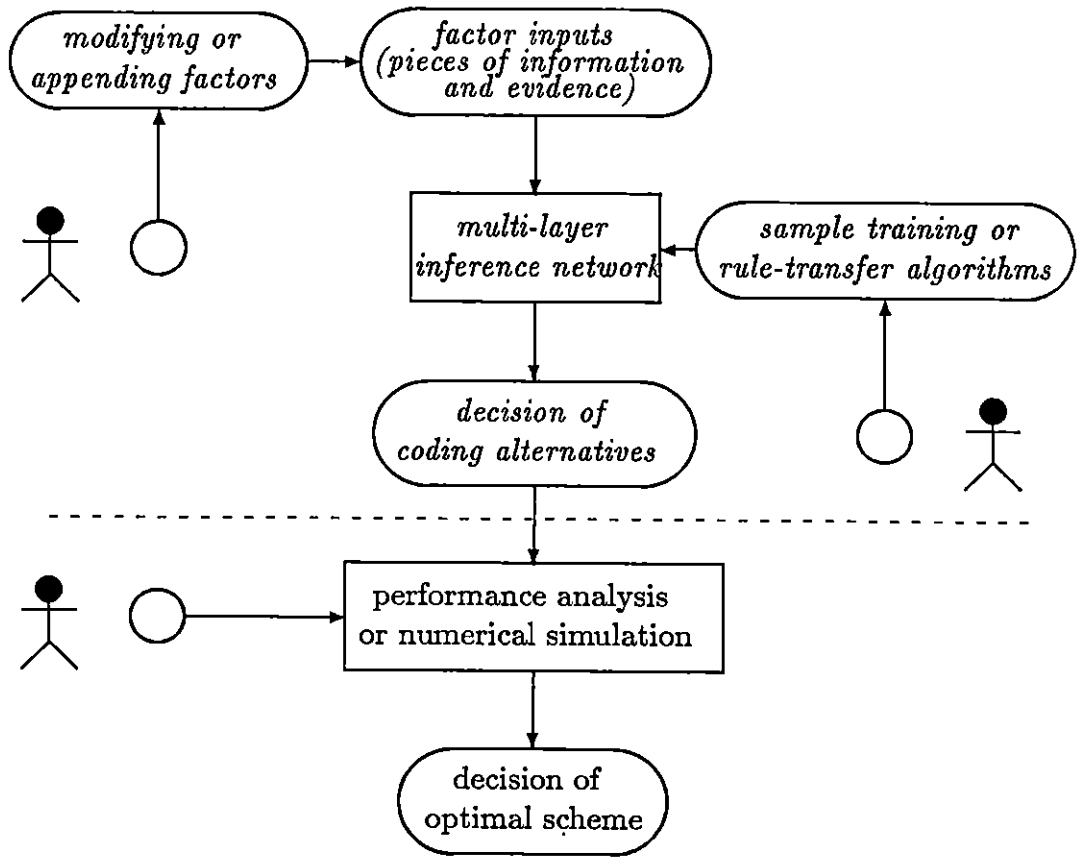


Figure 3.1: Framework of a Knowledge-combined Coding Selection System

3.4 A Knowledge-Combined Approach

Now we are ready to propose and define what is "a knowledge-combined approach".

A knowledge-combined approach is a problem-solving method by an expert system which has the ability to combine qualitative characteristics and quantitative models in the same framework to draw conclusions.

The structure of a knowledge-combined expert system for the coding selection problem is outlined in Figure 3.1. The lower part contains the quantitative analysis which comprises conventional engineering methods.

The components in the ovals are the input or output blocks, and the circles represent the human-machine interfaces.

The upper part in Figure 3.1 is built on a knowledge-combined inference network which is discussed in the next chapter. The systematic procedures for developing such an inference network for coding selection can be described as follows:

Step 1: Construct an inference network shell whose parameters are to be determined.

Step 2: Determine all the factors, F_1,F_2,..., F_k, which may affect a coding selection problem. When necessary, modify existing factors or append new factors.

Step 3: Determine feasible values of F_i $(i = 1, 2, ..., k)$. For the factors with qualitative uncertainty, assign a number ranging from 0 (least certain) to 10 (most certain). For the factors with quantitative models, assign 0 corresponding to the minimum, 10 to the maximum and quantify the value to $0, 1, 2, ..., 10$.

Step 4: Prepare as many as possible practical coding selection samples or inference rules.

Step 5: Input all the factor values of an available sample or rule.

Step 6: Train the inference network by using the rule-transfer algorithms described in the next chapter.

Step 7: If more samples or rules are available, then go to Step 5, otherwise stop training.

Step 8: When necessary, refine the inference network by using rule-transfer algorithms.

In the above procedures, we have combined the characteristics of quantitative and qualitative factors in Step 3, Step 6, and Step 8. Also, as shown

in Figure 2.1, we have combined a knowledge inference network and performance analysis or numerical simulation in the same framework. Thus, our method is a knowledge-combined system approach.

3.5 Summary

- Rules can be used to represent expert knowledge for coding selection, but rules are not capable of capturing the full range of knowledge for engineering system design of error control coding.
- A knowledge-combined approach is a problem-solving method by an expert system which has the ability to combine qualitative characteristics and quantitative models in the same framework to draw conclusions.

Chapter 4

Construction of an Inference Network with Rule-Transfer Algorithms

4.1 On Probabilistic Inference Networks

As a formal structure for representing decision-making systems, the probabilistic inference network is known to be good at handling information processing tasks with the following characteristics [55]:

- pieces of information are available at various levels of certainty and completeness;
- there is a need for optimal or nearly optimal decisions;
- general rules of inference are known or can be found for the problem.

In Figure 3.1, we have seen that the proposed system can be built on an inference network. The more detailed inference network is shown in Figure 4.1. This is a modification of the conventional probabilistic inference network model which has been successfully used in building expert systems such as PROSPECTOR [24].

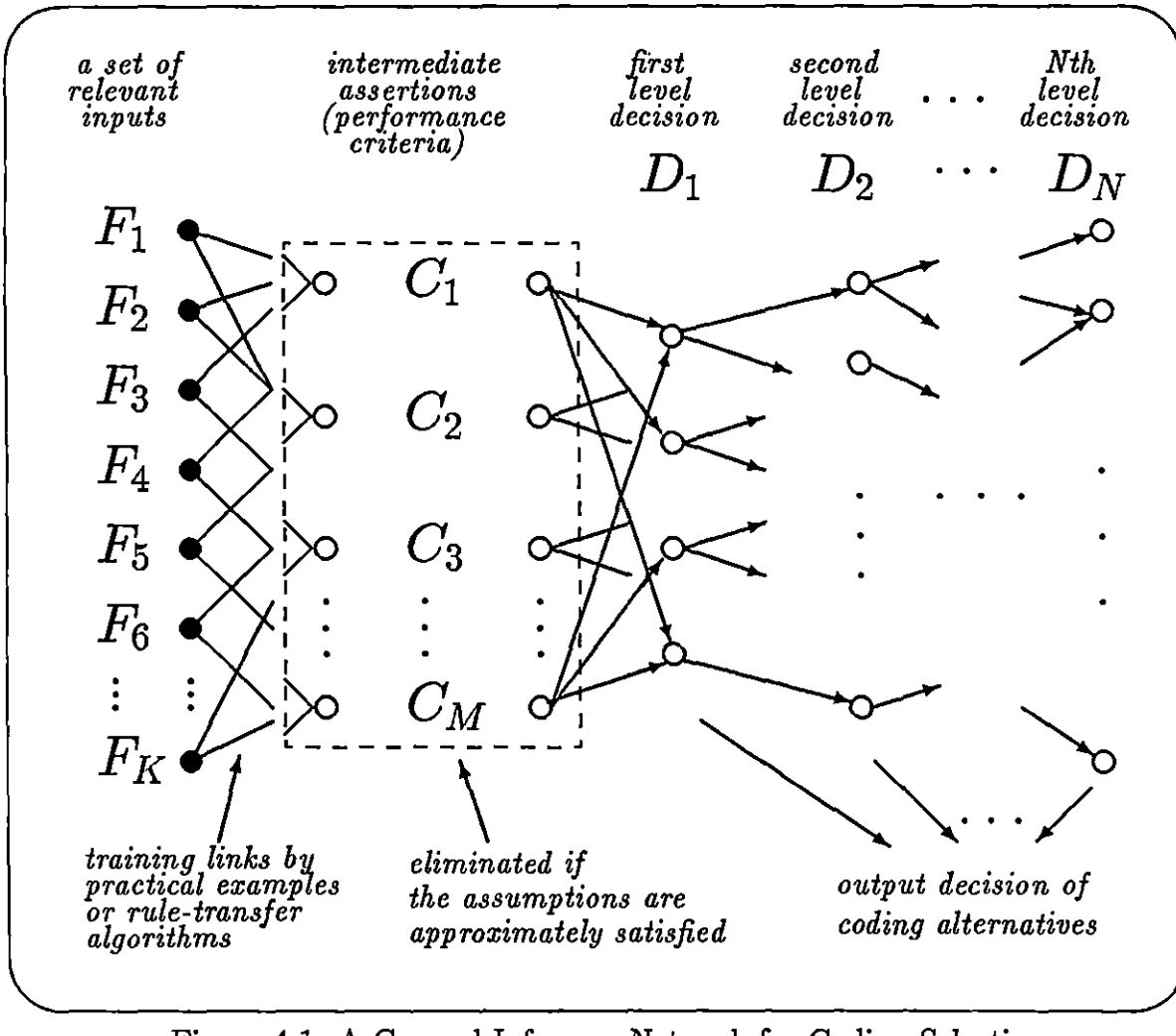


Figure 4.1: A General Inference Network for Coding Selection

The conventional probabilistic inference model is based on the traditional Bayesian approach (maximum likelihood method) [55]. But the model has the following disadvantages:

- Its assumption on conditional independence almost never holds in practice.
- There is no simple method available for building and refining the inference network.
- In order to obtain the probabilities in the network, a large number of statistical samples are required. This may not be possible for a particular application.

Despite these problems, many successful expert systems have been built on the basis of the Bayesian model with some modifications.

4.2 Derivation of a Perceptron-Based Inference Network

Our approach is going to incorporate the advantages of rule-based inference into the Bayesian model to compensate for both conventional approaches. This makes the inference network model more suitable for our coding selection problem.

A brief derivation for our modified probabilistic inference network is described as follows.

In Figure 4.1, each training link is implemented by a decision function. If the independence assumption is approximately satisfied by carefully choosing the factors affecting coding selection, then the intermediate assertions in Figure 4.1 can be eliminated and the model works well. This may not be a normal case. In general, it is much easier to choose the factors that are either approximately independent or highly dependent on each other.

Let a set of relevant input factors be denoted by F_1, F_2,..., F_k. Consider any decision level, say first decision level, which has a set of feasible output decisions denoted by D_{11}, D_{12},..., D_{1l}. We want to find an inference model which is able to make decisions from any given set of input factors. This can be formulated as to find a set of decision functions f_{ij} such that

$$D_{1j}^{*} = f_{1j}(F_1, F_2, ..., F_k) \quad j = 1, 2, ..., l \tag{4.1}$$

and the decision rule is then determined from the relative values of the calculated D_{1j}^{*} $(j = 1, 2, ..., l)$.

Consider general cases where F_i $(i = 1, 2, ..., k)$ are not independent of each other. Let all the dependent F_i's be members of C_i, i.e., any C_i (i=1,2,..., m) only contains dependent input factors and C_i's are independent of each other.

For the sake of simplicity, let $C_i = 1$ if C_i is observed , otherwise $C_i = 0$. Also define

$$p_j = P(D_{1j}),$$
$$p_{ij} = P(C_i = 1|D_{1j}),$$
$$q_{ij} = 1 - p_{ij} = P(C_i = 0|D_{1j}).$$

Suppose now that a given set of input factors is observed: $F_1 \in C_1$, $F_2 \in C_2$,..., $F_n \in C_n$ where $n < m$, we want to know which D_{1j} is most probable to be selected as system output. According to Bayes' theorem, we choose j with maximum likelihood function as

$$\begin{aligned} D_{1j}''' &= p_j \prod_{C_i=1} p_{ij} \prod_{C_i=0} q_{ij} \\ &= p_j \prod_i p_{ij}^{C_i} \, q_{ij}^{1-C_i} \\ &= p_j \prod_i \left(\frac{p_{ij}}{q_{ij}}\right)^{C_i} \prod_i q_{ij} \end{aligned} \tag{4.2}$$

Because only relative values of D_{1j}'s are interesting and sums are easier to calculate, we take logarithmic functions on both sides of equation (4.2). Thus,

$$D''_{1j} = \sum_i log C_i \frac{p_{ij}}{q_{ij}} + \left(log p_j + \sum_i log\ q_{ij} \right) \tag{4.3}$$

The second term in equation (4.3) is a constant that depends only on j. It is reasonable to omit this constant because we have no preference for deciding any D_{1j} without evaluating its performance. Let $a_{ij} = log \frac{p_{ij}}{q_{ij}}$, then

$$D'_{1j} = \sum_i^m a_{ij} C_i \tag{4.4}$$

Thus, the decision rule is as follows:

$$D_{1j}\ is\ output\ if\ D'_{1j} \geq D'_{1k},\ for\ all\ k \neq j$$

This decision rule is already known as the maximum-likelihood method and has been used as a basis of the probabilistic inference network model.

Now consider the case without the independence assumption. We note that it has been assumed in the above derivation that the occurrence of C_i is equivalent to the occurrence of the corresponding F_i. This is only true for the case where C_i contains one F_i. Consider $C_1 = \{F_1, F_2, ..., F_n\}$. Assume that C_1 and $F_i (i = 1, 2, ..., n)$ are linearly dependent (not conditional independent) on each other. We may write

$$C_1 = b_1 F_1 + b_2 F_2 + \cdots + b_n F_n \tag{4.5}$$

where b_i's are constants to be determined. Combining equations (4.4) and (4.5)

$$D_{1j}^{**} = \sum_{i=1}^{m}\sum_{k=1}^{n} a_{ij}b_{ik}F_{ik}$$

$$= \sum_{r=1}^{mn} c_r F_r \tag{4.6}$$

where $c_r = a_{ij}b_{ik}$, and $F_r = F_{ik} = 1$ if F_r is observed. For those F_r's which do not occur in a particular decision process, $F_r = 0$.

Further, in order to handle input factors with uncertainty, we use the fuzzy set concept [66] to extend the value of F_i as a membership number in the interval $[0, 1]$ and denote it as $M(F_i)$. The special case is $M(F_i) = F_i = 1$ for any observed F_i. In general, the higher the certainty of the input factor F_i, the larger the value that is assigned to $M(F_i)$. Based on this extension, let $K = mn$ and $w_i = c_i$, the final *decision function* can be written as

$$D_{1j}^{*} = \sum_{i=1}^{K} w_i M(F_i) \tag{4.7}$$

and the *decision rule* is as follows:

$$D_{1j} \textit{ is output if } D_{1j}^{*} \geq D_{1k}^{*}, \textit{ for all } k \neq j.$$

Similarly, we can obtain the same results for any other decision level in the inference network. It is interesting to note that the format of equation (4.7) is similar to the well-known pattern classification model using linear discriminant functions [16]. Our derivation is different from the conventional way in that we have considered not only the conditional independent case, but also the linear dependent case, and have extended the result to handle uncertain information. In the next section, we further modify the

inference model to include another important dependent case — exclusive disjunction.

Similar to other existing expert system models, in our inference formula (4.7), the values of $M(F_i)$'s can be determined by subjective assignment. But the values of weights (w_i) are to be determined by rule-transfer algorithms which are described in the following sections. These algorithms provide an efficient way to develop knowledge-combined expert systems.

4.3 Rapid System Prototyping Algorithms

During the past twenty years, artificial intelligence has achieved considerable success in the development of expert systems. But often an expert system takes a long time to build, even with the help of a domain expert. As we have mentioned before, maintenance and refinement of an existing expert system is also a complicated task. In our opinion, knowledge acquisition and refinement are the major difficulties in developing an expert system. Therefore, we propose a method named *Rule-Transfer Algorithms*. These algorithms are used in the period of system prototyping for knowledge acquisition and in the period of system maintenance for knowledge refinement.

In the system prototyping period, we have a set of general inference rules available, which may not meet the completeness requirement. Links in the inference network are implemented by decision functions as defined in equation (4.7). Given the basic types of inference rules, the links can be formed by using the following corresponding algorithms on the basis of correlations among attributes.

1. **Logical concurrence** — an input F_i is highly correlated with an output D_j;

   Algorithm 1: in the decision function of D_j, based on the degree of

the correlation, set a number between 0 and 1 to the corresponding w_i, e.g., $w_i = 0.75$.

2. **Negative concurrence** — strong negative correlation between F_i and D_j;

   Algorithm 2: in the decision function of D_j, set the corresponding $w_i = -0.75$.

3. **Logical implication** — whenever F_i occurs, D_j does too;

   Algorithm 3: in the decision function of D_j, set the corresponding $w_i = 1$.

4. **Logical inclusion** — whenever an input F_i occurs, another input F_j does too;

   Algorithm 4: simply discard F_i when computing decision functions.

5. **Conjunction** — output D_j occurs whenever both inputs F_i and F_k occur;

   Algorithm 5: in the decision function of D_j, set the corresponding $w_i = 0.5$ and $w_k = 0.5$.

6. **Disjunction** — output D_j occurs whenever either input F_i or F_k occurs;

   Algorithm 6: in the decision function of D_j, set the corresponding $w_i = 1$ and $w_k = 1$.

7. **Exclusive disjunction** — output D_j occurs whenever either input F_i or F_k occurs, but not both;

   Algorithm 7: in the decision function of D_j, set the corresponding $w_i = 1$ and $w_k = 1$ and modify the decision function of D_j by appending an additional term $-2M(F_i)M(F_k)$, i.e., equation (4.7) is modified as

$$D_{1j}^{*} = \sum_{i=1}^{K} w_i M(F_i) - 2 \sum_{i,k} M(F_i) M(F_k) \tag{4.8}$$

8. **Sufficient implication** — if all of F_{i+1}, F_{i+2},..., F_{i+k} occur, then D_j occurs too;

   Algorithm 8: in the decision function of D_j, set the corresponding $w_{i+1} = 1/k$, $w_{i+2} = 1/k$,..., $w_{i+k} = 1/k$.

9. **Necessary implication** — if input F_i does not occur, then D_j does not occur either;

   Algorithm 9: redefine input F_i by its inverse denoted as F_i', i.e., define $M(F_i') = 1 - M(F_i)$, and in the decision function of D_j, set the corresponding $w_i = -1$.

Of course, the above algorithms represent only one of many reasonable ways for assigning weights to inference links. It is easy to verify the correctness of the algorithms by introducing a default *unit* threshold in all decision functions. In the case of unequal importance of multiple inputs being realized, the weight distribution can be further modified to improve the initial system settings.

4.4 Knowledge Refining Algorithms

During the period of system refinement, if more inference rules are available, our rule-transfer training algorithm can be described as follows:

Rule-transfer Algorithm

Step 1—Set Initial Training Parameter: At training time $t = 0$, set training speed control parameter $\alpha(0) = 1$.

Step 2—Decompose Given Rule: Decompose the given inference rule into valued input factors F_i $(i = 1, 2, ..., k)$ and the desired output D_{nj} $(n = 1, 2, ..., N)$.

Step 3—Set Membership Numbers: For F_i with full certainty, set $M(F_i) = 1$; for F_i with uncertainty values $1, 2, ..., 9$, set $M(F_i) = 0.1, 0.2, ..., 0.9$, respectively; for F_i not occurring in the given rule, set $M(F_i) = 0$.

Step 4—Calculate Decision Functions:

$$D^*_{nj} = \sum_{i=1}^{K} w_i(t) M(F_i) \ , \quad j = 1, 2, ..., l, \ n = 1, 2, ..., N \tag{4.9}$$

Step 5—Determine System Outputs: Select D^o_{nj} as system outputs by finding the maxima of D^*_{nj} for every $n = 1, 2, ..., N$.

Step 6—Verify System Outputs: If D^o_{nj} are the same as the desired outputs D_{nj}, then *stop*, otherwise go to Step 7.

Step 7—Adapt Training Parameter: In order to adjust training speed, initially set $\alpha(1) = \alpha(0) = 1$. If the current system outputs $D^o_{nj}(t)$ are the same as $D^o_{nj}(t-1)$, then set $\alpha(t) = 2\alpha(t-1)$, otherwise set $\alpha(t) = 0.5\alpha(t-1)$.

Step 8—Adapt Weights: For those D^o_{nj} which are not desired outputs, the weights in the corresponding decision functions are adapted as follows:

$w_i(t+1) = w_i(t) - \delta$, for all i corresponding to a given input F_i, other weights in the function remain unchanged, where

$$\delta = \alpha \frac{\sum_{i=1}^{k} w_i}{k} ; \tag{4.10}$$

for those D^o_{nj} which are desired outputs, the weights in the corresponding decision functions are adapted as follows:

$w_i(t+1) = w_i(t)+\delta$, for all i corresponding to a given input F_i, where δ is the same as equation (4.10), and other weights in the function remain unchanged.

Step 9—Iteration: Using the same given inference rule, set $t = t + 1$ and repeat by going to Step 4.

For system refinement using practical observations as training samples, the above algorithms can also be used, by simply replacing 'rules' by 'samples'. Since samples may be noisy, we should use a smaller adaptive value in Step 8, e.g., set $\delta_{sample} = 0.1\delta_{rule}$ to make the model relatively stable.

The above rule-transfer algorithm for system refinement has a similar structure with linear discriminant function or perceptron model, which has been shown to be convergent for linearly separable samples [16] [39].

It is important to point out that the weights in the decision functions are relatively stationary and they are not required to be stationary all the time, because as the knowledge of the coding experts grows and changes, new inference rules may not be consistent with the old rules used for training the existing system. For example, since 1960 the cost of digital electronics have been decreasing at a phenomenal rate. Thus, conclusions on factors such as speed, cost, weight or power in coding selection based on the technology ten years ago may be either irrelevant or no longer valid due to new technology.

For multi-expert case, the inference rules may not be perfectly consistent with each other. In such a situation, we may treat these rules as a set of training samples with noisy components, and repeat the rule-transfer algorithm many times until they are consistent in the same inference model. If system convergence cannot be achieved in this case, exclusive disjunction or other nonlinear relationships should be investigated and realized.

Theoretically, it is difficult to prove when and how fast the network converges to a correct state. This remains an open problem to be studied further.

4.5 On System Implementation

Lisp or C can be used to implement the proposed knowledge-combined expert system. We may also consider the use of well-known expert system tools such as Knowledge Craft and KEE. But in our approach we need to integrate numerical models into the expert system. The commercial development tools seem too expensive in both product price and computing resource requirements.

For system prototyping, the C language has been chosen to implement the coding selection expert system. At present, we only consider n known coding schemes as feasible system outputs. Each coding scheme has its own decision function. The system is trained using the rule-transfer algorithms. The decision of output coding alternatives is then made by decision rule as described in the previous section.

As an example, the system is trained by using the inference rules including the results in [42]. The following interactive dialogue session demonstrates the computer-aided coding selection by our knowledge-combined expert system.

```
Sample Session
```

A Knowledge-Combined Coding Selection Expert System

```
Please answer the following questions...
Do you prefer some types of coding strategy for your problem?
```

Respond with yes or no.

> no

OK!

All the available coding schemes in this system are to be evaluated for your application.

You may answer the following questions
with a yes(Y), no(N), maybe(M), or simply don't know(D),
but the system prefers an answer with a probability factor
ranging from 0 (least certain) to 10 (most certain).

If you don't understand the meaning of the questions, type help(H).

Do you require a very low decoded bit error rate?

> H

The decoded bit error rate means the allowed rate of accepting incorrect messages. This rate may range from 10^{-10} to 10^{-5}.

Again, do you require a very low decoded bit error rate?

> 8

Do you require a constant throughput rate to transmit messages.

> Y

Do you have a communication channel with only random errors?

> M

The available channel bit error rate is very high?.

> D

Do you have a communication feedback channel available ?

> Y

Do you have a block structure of message format ?

> Y

Do you have very long channel round trip time delay ?

> H

The typical round trip time delay is ranging from a few milliseconds to as high as 1000 milliseconds.

> 7

. . . . . .

Do you have a Viterbi decoder available?

> no

OK! Your information is being processed.

Please wait . . .

It is done!

Our knowledge-combined coding selection expert system suggests that you use the following coding scheme:

This is a rate-constant Type-I hybrid ARQ/FEC scheme with BCH error correcting codes.

This scheme is best suitable for your application because...

1. a feedback channel is available;
2. a constant throughput and a constant delay are required;
3. the channel bit error rate varies through a wide range;
4. a high system reliability is required.

Do you need more information? Yes(Y) or No(N)...

> Y

The research shows that the suggested scheme with a binary BCH code requires less E_b/N_0 than an FEC scheme using a convolutional code and soft decision Viterbi decoding.

Please type continue(C) for further simulation results and parameter optimization for your scheme, otherwise type stop(S).

> S

Bye now !

Numerical simulation and optimization results for the above example is described in Chapter 6.

4.6 Summary

In this chapter, we have provided a framework for developing a knowledge-combined expert system for optimal error control coding selection. Two important issues for developing such a system have been identified. One concerns construction of knowledge base and inference mechanism on a modified probabilistic inference network basis, and the other concerns knowledge acquisition by sample or rule training by rule-transfer algorithms.

By using a knowledge-combined approach, conventional expert system methods are improved to be more suitable for solving engineering design problems. Like other successful expert systems, the knowledge-combined system approach described in this chapter has the ability to handle symbolic information and the ability to accommodate new expertise whenever some new knowledge is identified by an expert. Furthermore, the proposed rule-transfer algorithms are designed to be able to make rule-based expert systems easier to be built and refined.

Chapter 5

Performance Analysis for Type-I Hybrid ARQ Schemes

5.1 Introduction

The hybrid scheme may have a higher reliability than a pure forward error correction and a higher throughput than a system with retransmission only. Hybrid ARQ schemes are basically classified into two categories. In a Type-I hybrid ARQ scheme an inner code for error correction and an outer code for error detection are used or simply a single code is designed for simultaneous error correction and detection [47]. A typical Type-II hybrid ARQ scheme [14] uses separable codes where only parity bits for error detection are sent on the first transmission so that the upper limit on throughput efficiency is near 1. Throughput suffers only when retransmissions are required, then parity bits for error correction are sent.

The performance of a hybrid ARQ error control scheme is characterized by throughput efficiency, time delay, and undetected error probability. In this chapter, we use the ARQEX scheme [63] to discuss the issues of performance analysis for Type-I hybrid ARQ schemes.

5.2 A Proposed Hybrid ARQ Error Control Scheme

In the communications context, both ARQ and FEC techniques add redundancy to data prior to transmission in order to reduce the effect of errors that occur during transmission, although the philosophy is very different. Here we emphasize that all the previous error-control schemes are based on appending redundancy to data, but the redundancy in real data is never used for error-correcting purpose. It may be very complicated to use the redundancy of real data in the general sense. However the data in a text document, which may be written in a natural language, contain great redundancy so that one or two errors in a sentence do not lead to to misunderstanding. On the other hand, it is known that real errors may be classified as burst errors and non-burst (or random) errors. In the random error case, if one or more errors occur in a block, entire data is retransmitted. The data user most likely does not care whether a block has one or 100 errors, as it is discarded in conventional ARQ schemes.

Obviously, if we could *use the redundancy of real data in error-control schemes, not by way of data compression when source encoding*, the system performance could be much improved. In practice, the case of document transmission provides the possibility to use the redundancy in data. We define a document as the file containing the contents written in a natural language. This is often the case in office automation networks where most files are created by word processing software and electronic-mail tools are used to transmit files in the networks.

In fact, a more general case is the file transfer in some applications where two nodes of communications have the same available dictionary. All data in the files can be found in the dictionary. In such practical situations, if the received files contains a few errors, based on the available dictionary, we may

simply use an off-line spelling-correcting system to obtain fine document output and improve throughput efficiency.

An actual file may contain data which cannot be found in the available dictionary and the spelling-correction may fail in the the case where one received word in error corresponds to a few similar words in the dictionary. In this situation, a decision making mechanism can be considered.

The general block-diagram of the proposed scheme, denoted in the following with ARQEX, is shown in Figure 5.1. For simplicity, we assume that user A transmits files to user B. Note that five new components are introduced into the proposed system. They are Data Separator (DS), Spelling Corrector (SC), Spelling Decision (SD), Data Reconstructor (DR), and Dictionary (DD). All of these can be implemented by software on the node computer which is readily available in point-to-point data communications, where Station A and Station B can transmit data files to each other. We may call the software system implementing the above five components as EX for the sake of simplicity. Because the EX system is implemented on the node computer, the impact of EX on the system cost is negligible. There is no impact on transmission delay, since EX, which is running in an off-line manner, does not occupy transmission lines.

In current office automation environments, powerful word processing software with spelling-checking function is widely used. It can be expected that robust spelling correcting systems will be developed in the near future [17].

The only condition to use the ARQEX scheme is that there exists the same dictionary DD in both Station A and Station B. DD contains finite words which are understandable to the users in both stations. An actual data file F is composed by words which may or may not be found in the predefined DD. The DS module is used to separate the contents of F into two parts: part W, in which all the words can be found in DD, or $w \in$ DD if $w \in W$, and part XW, in which all the words cannot be found in

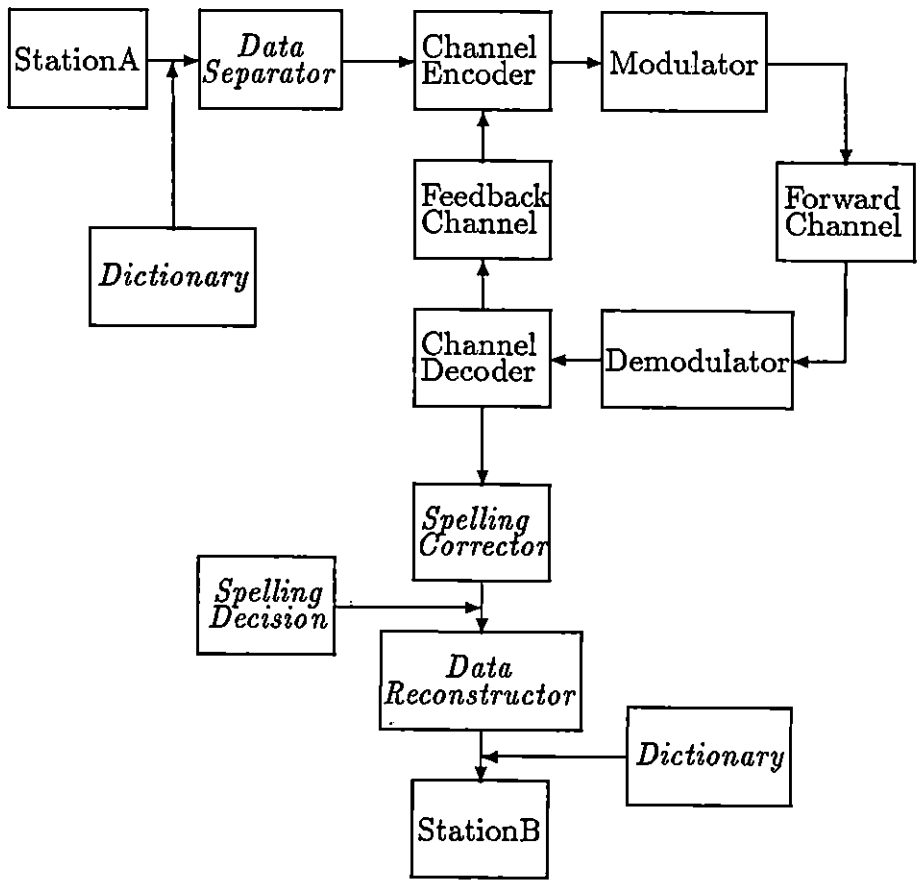


Figure 5.1: Basic Blocks of the Proposed Hybrid ARQ Scheme

DD, or w $\ni$ DD if w $\in$ XW. All the words are composed of symbols. The node computer generates symbols from a finite alphabet $S = \{a_1, a_2, ..., a_M\}$ with M elements. Each block of k information symbols coming out of DS is encoded with a code C_o of type (n, k). Code C_o is assumed to be able to detect s errors.

The part XW and the part W of F are transmitted in separate frames. For the part XW, the transmission mechanism is the same as the basic Go-Back-N ARQ scheme. When the part W is to be transmitted, the system switches to the protocol in which the retransmission occurs only if the detected error number $e > k$, where $k < s$. This implies that detected k or less errors are corrected in the SC and SD modules by using the redundancy in the data of W.

In time of only non-burst errors occurring, we may simply transmit the part W without retransmission. Code C_o can then be easily chosen and the random errors occurring in the part W are to be corrected by SC and SD.

Most of the errors in received part W can be corrected in SC by simply using checking the spelling according to DD. If the received word in error corresponds to more than one word in DD, then SD is used to make a decision. SD could be an expert system based on the statistical pattern classification or other available inference models [22]. SD is easily implemented if we leave it to be trained while the system is running. In this case SD works in an interactive mode and may be used to monitor the error patterns in actual transmission at the same time.

When all the errors are corrected by the SC and SD modules, the received file is reconstructed by the DR module. This means that the part W and the part XW are rearranged into the original transmitted file.

It is noted that when F is divided into W and XW, in order to reconstruct F, we must include the additional address bits in XW. For simplicity, we take the number of the address bits equal to the number of the bits in the part XW. Before DS we may write

$$N_F = N_W + N_{XW} \tag{5.1}$$

where N_F, N_W, and N_{XW} stand for the numbers of bits in F, W, and XW, respectively.

After DS we have

$$N'_F = N_W + 2N_{XW} \tag{5.2}$$

where N'_F stands for the number of bits corresponding to the original F after data separation by DS. Note that $N'_F > N_F$, but $N_W \gg N_{XW}$ so that in the proposed ARQEX scheme N_W bits are transmitted with much less or without retransmissions compared with classical ARQ schemes. Thus, the throughput efficiency is improved, particularly for the applications of text document transmission. Another advantage of the ARQEX scheme is that the undetectable errors, which are not be corrected in any existing ARQ schemes, may be easily corrected in our ARQEX scheme because the functions of SC and SD modules.

5.3 Throughput Analysis

In this section we derive the throughput performance of the proposed ARQEX scheme using a similar approach as adopted for the basic Go-Back-N ARQ scheme. In order to focus on the key points, we consider the general idealized assumptions [49]. Sequence numbers in frames are assumed to be as large as possible. We also assume fixed-length data frames; a constant, known round-trip propagation delay between transmitter and receiver; and a fixed, known processing delay at the receiver.

Let t_I be the time required to transmit a frame (data packet plus control bits). Let t_{out} be the timeout interval, at the end of which an acknowledg-

ment (ACK) arrives or a retransmission takes place. Let $t_T = t_I + t_{out}$ and $a \equiv t_T/t_I \geq 1$. To determine the maximum throughput we shall assume that frames are transmitted from A to B only, with B replying with an ACK or a negative acknowledgment (NACK). Transmitter A is further assumed to be operating in a saturated state which means there always exists a frame waiting for transmission. The expression to be found thus represents an upper limit on the throughput performance.

Suppose that the probability of a part XW frame being received in error at station B is P_{XW}, and the probability of a part W frame being received in more than k errors at station B is P_W. Since we assume $N'_F = N_W + 2N_{XW}$, the average time for a successful transmission is given by

$$\begin{aligned}
t_V &= \frac{2N_{XW}}{N_W + 2N_{XW}}[t_I + (1 - P_{XW})\sum_{i=1}^{\infty} iP_{XW}^i t_T] \\
&\quad + \frac{N_W}{N_W + 2N_{XW}}[t_I + (1 - P_W)\sum_{i=1}^{\infty} iP_W^i t_T] \\
&= t_I + \left[\frac{2N_{XW}(1 - P_{XW})\sum_{i=1}^{\infty} iP_{XW}^i + N_W(1 - P_W)\sum_{i=1}^{\infty} iP_W^i}{N_W + 2N_{XW}}\right] t_T \\
&= t_I + \left[\frac{2N_{XW}\frac{P_{XW}}{1-P_{XW}} + N_W\frac{P_W}{1-P_W}}{N_W + 2N_{XW}}\right] t_T \\
&= t_I\left\{1 + \frac{a[2N_{XW}P_{XW}(1 - P_W) + N_W P_W(1 - P_{XW})]}{(N_W + 2N_{XW})(1 - P_{XW})(1 - P_W)}\right\} \\
&= t_I\left\{\frac{N_W(1 - P_{XW})[1 + (a-1)P_W] + 2N_{XW}(1 - P_W)[1 + (a-1)P_{XW}]}{(N_W + 2N_{XW})(1 - P_{XW})(1 - P_W)}\right\}
\end{aligned} \tag{5.3}$$

Under the saturation assumption, the maximum possible throughput is given by

$$\lambda_{max} = 1/t_V \tag{5.4}$$

Let the length of the packet (data) field in the frame be l bits, with a total of l' bits used in the remaining (control) fields. Then the average data rate, in bits/sec of data delivered to Station B is

$$D = \lambda_{max} l \tag{5.5}$$

Further let $t_I = (l+l')/C$, where C is the link transmission rate capacity in bits/sec, from equations (5.3),(5.4), and (5.5) we have, for the normalized data rate,

$$D/C = \left(\frac{l}{l+l'}\right) \frac{(N_W + 2N_{XW})(1-P_{XW})(1-P_W)}{N_W(1-P_{XW})[1+(a-1)P_W] + 2N_{XW}(1-P_W)[1+(a-1)P_{XW}]} \tag{5.6}$$

Note that when $N_W = 0$ (all the received data in error to be retransmitted based on Go-Back-N protocol), then $P_W = 0$, from equation (5.6) we have the same result as the basic GBN scheme, i.e.,

$$D/C = \left(\frac{l}{l+l'}\right) \frac{1-P_{XW}}{1+(a-1)P_{XW}} \tag{5.7}$$

Now consider the special case for which we take $P_W = 0$ but $N_W > 0$. This corresponds to all the random errors in part W to be corrected in SC and SD modules, or no retransmission for part W. Then we have

$$D/C = \left(\frac{l}{l+l'}\right) \frac{(N_W + 2N_{XW})(1-P_{XW})}{N_W(1-P_{XW}) + 2N_{XW}[1+(a-1)P_{XW}]} \tag{5.8}$$

In practical documents, the typical case may be $N_{XW} = 0.05N_W$, which means 5% of the data in the document, such as digital variables, cannot be found in the predefined dictionary DD. In this case we have

$$D/C = \left(\frac{l}{l+l'}\right)\frac{1-P_{XW}}{1+(\frac{a}{11}-1)P_{XW}} \tag{5.9}$$

Another special case is that $N_{XW} = 0$, which corresponds to the case where all the data in the document are included in the predefined dictionary DD. Then we have

$$D/C = \left(\frac{l}{l+l'}\right)\frac{1-P_W}{1+(a-1)P_W} \tag{5.10}$$

In this case, we make the assumption that bits are independently prone to error, with a bit error probability P_b. This assumption turns out to be valid in the case where errors are due primarily to random noise. It is apparent that the frame error probability (the probability that at least one bit is received in error) is given by

$$P = P_{XW} = 1-(1-P_b)^{l+l'} \tag{5.11}$$

Let P_k be the probability of k errors in a frame. P_k is equal to $P_b^k(1-P_b)^{l+l'-k}$ if the errors are in specified locations. If errors can be in any possible location, as is typically the case, the result becomes

$$P_k = \left(\begin{array}{c} l+l' \\ k \end{array}\right) P_b^k(1-P_b)^{l+l'-k} \tag{5.12}$$

In the case of large error probabilities (such as for satellite links), say $P_b = 10^{-3}$, then one error probability in a frame $P_1 = 0.37$, and

$P_1 + P_2 + P_3 = 0.61$ when $l + l' = 1000$ bits, which is a typical value in practice. Obviously in the frame of part W, if $l + l' = 1000$ bits, one or two bits in error are easily corrected by simply using the SC module. Thus, the throughput performance can be much improved in the case of high error rate channels. From equations (5.11) and (5.12), we have the part W retransmission probability (the probability that more than k bits received in error) as follows

$$\begin{aligned} P_W &= P - \sum_{i=1}^{k} P_i \\ &= 1 - (1 - P_b)^{l+l'} - \sum_{i=1}^{k} \binom{l+l'}{i} P_b^i (1 - P_b)^{l+l'-i} \end{aligned} \tag{5.13}$$

Further consider the special case where retransmissions occur only if more than one error is detected in the frame of part W, then equation (5.13) becomes

$$\begin{aligned} P_W &= P - P_1 \\ &= 1 - (1 - P_b)^{l+l'} - (l + l')P_b(1 - P_b)^{l+l'-1} \\ &= 1 - (1 - P_b)^{l+l'} \left[1 + \frac{(l + l')P_b}{1 - P_b}\right] \end{aligned} \tag{5.14}$$

By now we have obtained the general expressions (5.6) and (5.13), and in the special interest cases we have the results of equations (5.9), (5.11) and (5.10), (5.14). All these expressions show clearly the effect on the data transmission rate of packet length l, control field length l', and bit-error probability P_b. The timeout interval is represented by the normalized parameter $a \equiv t_T/t_I = 1 + t_{out}/t_I$.

5.4 Comparisons of Throughput Efficiency

The three basic schemes which are used in ARQ techniques are: Stop-And-Wait (SAW), Go-Back-N (GBN) and Selective Repeat (SR). Many different modifications of these basic schemes have been proposed to increase their efficiency [9]. In this section, the basic ARQ schemes and some representative modified ARQ schemes as well as the proposed ARQEX schemes are outlined, and their throughput results are summarized. These results are then used to compare the throughput efficiency of the different schemes.

1. Stop-And-Wait

   The transmitter sends a codeword to the receiver and waits for an ACK from the receiver. If an ACK is received, the transmitter sends the successive codeword, while, if a NACK is received, the transmitter retransmits the codeword and waits for the successive ACK. The SAW scheme is quite simple to implement, but it is also inherently inefficient owing to the idle time spent in waiting for the ACK after each transmission. The throughput T is given by

   $$T = \frac{1 - P}{a} \tag{5.15}$$

   where P and a were defined as in the previous section.

2. Basic Go-Back-N

   The transmitter sends codewords continuously, one after the other. When a NACK is received, the transmitter interrupts the transmission of new codewords, backs up to the codeword negatively acknowledged and retransmits this codeword and the successive $N - 1$ codewords. At the receiver, the $N - 1$ frames following an erroneously received codeword are discarded, regardless of whether they contain

errors or not. The basic GBN scheme is more efficient than the SAW scheme, and presents a satisfactory performance when the error rate is not too high and the round-trip delay is small, but it becomes rapidly inefficient for high error rates and/or large round-trip delays. The throughput is given by

$$T = \frac{1-P}{1+(a-1)P} \tag{5.16}$$

3. Ideal Selective-Repeat

   The transmitter repeats only those frames which are detected in error, and hence it is more efficient than the other ARQ schemes. However, the implementation of the SR scheme requires an extensive buffer (theoretically an infinite buffer) at the receiver side, in order to deliver to the user the informative frames in a correct way, and more complex circuits are necessary. The throughput of the ideal SR scheme is given by

$$T = 1 - P \tag{5.17}$$

4. Sastry's GBN [48]

   In this modification of the basic GBN scheme, the transmitter enters a stutter mode as soon as a NACK for a frame is received, i.e., the transmitter keeps sending the negatively acknowledged frame (say ith) continuously until an ACK for ith frame is received. At this point the transmissions of frames $i+1$th, $i+2$th, etc. are resumed. For this scheme the throughput is

$$T = \frac{1-P}{1+2P(1-P)(a-1)} \tag{5.18}$$

5. Morris' GBN [41]

This is an enhancement of Sastry's GBN scheme with a buffer of size N frames in the receiver to save the good frames following the corrupted one. Here, the throughput is increased to

$$T = \frac{1-P}{1+P(1-P)(a-1)} \tag{5.19}$$

6. Moeneclaey-Bruneel's GBN [40]

When a particular data frame is to be transmitted to the receiver, the transmitter continuously sends a number of copies of this frame. Whenever a NACK for one of these copies is received, a new copy of the same frame is sent, until, eventually, an ACK occurs. The transmission of the next data frame can then be started, according to the same rule. This scheme yields a better throughput efficiency—especially for channels with large round-trip delay—than the other GBN schemes, for all frame error probabilities larger than 50%. The throughput is given by

$$T = \frac{1-P}{1+(1-P)(a-1)} \tag{5.20}$$

7. The proposed ARQEX scheme

The data to be transmitted are preseparated into the frame of part XW (N_{XW} bits) and the frame of part W (N_W bits). The transmitter sends XW frames in the same way as basic GBN scheme, but when the transmitter sends W frames, only if the received W frame contains more than k bits in error then the retransmissions are required. The normalized throughput, from equation (5.6), is given by

$$T = \frac{(N_W + 2N_{XW})(1-P)(1-P_W)}{N_W(1-P)[1+(a-1)P_W] + 2N_{XW}(1-P_W)[1+(a-1)P]} \tag{5.21}$$

where P_W is given by equation (5.13).

8. ARQEX in special case 1

   In this case, we are assuming only non-burst (random) errors may occur in the transmission channel, the transmitter sends W frames without retransmission. We take the typical value of $N_{XW} = 0.05N_W$, then the normalized throughput, from equation (5.9), is given by

$$T = \frac{1-P}{1+(\frac{a}{11}-1)P} \tag{5.22}$$

9. ARQEX in special case 2

   In this case, both Station A and Station B transmit messages based on the predefined dictionary DD such that no N_{XW} frames are required for transmission. Further we take the simple condition for implementation that only the received frames containing more than one bit in error are required to be retransmitted. Then the normalized throughput, from equation (5.10), is given by

$$T = \frac{1-P_W}{1+(a-1)P_W} \tag{5.23}$$

   and from equation (5.14) we have P_W given by

$$P_W = 1-(1-P_b)^{l+l'}\left[1+\frac{(l+l')P_b}{1-P_b}\right]$$

In all the above results, $P = 1-(1-P_b)^{l+l'}$, and the normalized data rate is

$$D/C = \left(\frac{l}{l+l'}\right)T \tag{5.24}$$

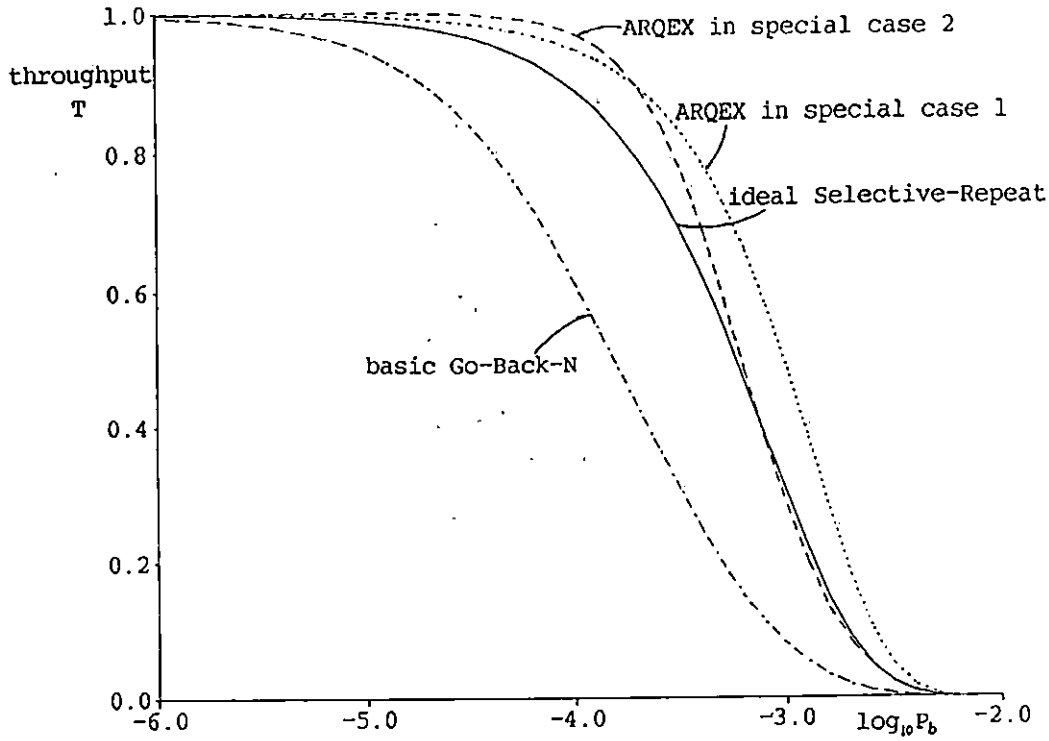


Figure 5.2: Relative Data Throughput versus Bit Error Rate

We can use these results to compare the throughput efficiencies of different schemes by plotting curves for typical cases. The relative throughput dependence of four strategies, i.e., the basic GBN, the ideal Selective-Repeat and the proposed ARQEX in case 1 and case 2, on the bit-error probability is shown in Figure 5.2. Here a typical fixed packet length of $l + l' = 1200$ bits (say $l' = 48$ bits) has been chosen, and the normalized data throughput (throughput efficiency) of the four schemes is plotted as a function of P_b. Equations (5.11), (5.14), (5.16), (5.17), (5.22), and (5.23) have been used for this purpose. A satellite channel has been assumed, with the typical propagation delay $t_p = 250$ msec. Taking $t_{out} = 2t_p + 2t_I$ and neglecting processing delay for simplicity. We then have the link parameter $a = 5$. Figure 5.2 clearly illustrates that the throughput efficiencies have been much improved by using the proposed ARQEX scheme. The interesting and significant point to note is that it is possible for ARQEX schemes to have better throughput even than that of ideal Selective-Repeat ARQ. This is because often fewer data frames, when errors occur, are required to

be retransmitted in ARQEX schemes.

5.5 Time Delay Analysis by Queueing Theory

The time delay analysis of ARQ schemes is more difficult than the throughput analysis. Delay and queue length evaluations are often omitted in papers dealing with the performance analysis of ARQ schemes. In practice, the queue length is one of the most prominent characteristics, because it enables the protocol designer to estimate the necessary buffer capacity, in order to avoid frame losses. Delay is also one of the key factors which affect the system cost, because more delay corresponds to more time occupying the transmission lines. Towsley and Wolf [58] analyzed the delay and queue length of the Stop-And-Wait and the Go-Back-N schemes. Another approximate delay analysis method[1] was applied to analyze the Selective-Repeat ARQ scheme. In this section, in order to compare the time delay of the proposed ARQEX scheme with the delay of the conventional ARQ schemes, we use a basic methodology similar to the analysis of the M/G/1 queue [23]. In this method, a Markov chain whose state is the number of messages in the system is embedded at the appropriate points. The solution to the steady-state equilibrium distribution of the chain allows one to find the average number of messages in the system. The application of Little's formula yields the average delay.

In practice, ARQ techniques are used for error control in asynchronous time-division multiplexing (ATDM) systems where a large number of users are multiplexed onto the same line. We assume that the aggregate arrival process to the multiplexer from all sources is λ messages per second. We also assume that the transmission rate on the synchronous line out of the buffer is $1/T_s$ slots per second, where a slot carries exactly one data unit. Consider the general case where a message may contain a number of data

units. Let us assume that the generating function of the number of data units in a message is expressed as $M(z)$. The receiver may detect errors in the data units. Again we use P to denote the probability of the event that a data unit is detected to be in error. We shall assume that errors on successive trials are independent events. We assume that there is no error in the feedback channel. Before transmission newly arrived data units are stored in a buffer. The first question that we shall address is the size of the buffer.

Let n_i be the number of data units in the buffer at the end of the ith slot. We embed a Markov chain at the time points between slots and assume the buffer is infinite. Let a_i denote the number of data units that arrive during the ith slot. Since at most one data unit is transmitted we have

$$n_{i+1} = n_i - U(n_i) + a_{i+1} \tag{5.25}$$

where $U(*)$ is the unit step function. The data units arriving during a slot cannot be transmitted until the beginning of the next slot. By taking the generating functions of both sides of equation (5.25), we can obtain the probability generating function of the system state, denoted by $P(z)$, as

$$P(z) \equiv E[z^n] = \frac{(1-\rho)(1-z)A(z)}{A(z)-z} \tag{5.26}$$

where $A(z)$ is the probability generating function for the number of data units arriving in a slot interval (T_s seconds). The procedures leading to the equation (5.26) and the conditions for a steady-state solution are basically the same as those for the M/G/1 queue [28]. As in the case of the M/G/1 queue ρ is the system load and is given by

$$\rho = \lambda T_s m \tag{5.27}$$

where m is the average number of data units in a message. For a steady-

state solution to exist we must have $\rho < 1$. By clearing fractions in equation (5.26) and differentiating successively, we have the average number of data units in the system given by

$$n = \frac{(1-\rho)A'(1)}{1-A'(1)} + \frac{A''(1)}{2[1-A'(1)]} \tag{5.28}$$

Because of the probability of error in the channel, the embedded points are not separated by single slot times. Let us consider the conventional ARQ schemes which are based on the continuous transmission or the Go-Back-N technique. In this case data units are transmitted in every slot. When a data unit is detected to be in error, the erroneous frame as well as all succeeding data units up to the time of the reception of the NACK are retransmitted. In the absence of error the time to transmit a data unit is a single slot. Let R denote the round-trip delay in slots. Since the first transmission takes only one slot, but all subsequent retransmissions take $R+1$ slots, if there are k transmissions the time required to transmit a data unit is $1+(k-1)(R+1)$. For simplicity we assume that a data unit leaves the system as soon as a successful transmission is completed. Actually it remains until an ACK is received.

If messages arrive at a Poisson rate of λ messages per second the probability of l messages arriving in the interdeparture interval is

$$P(l/k\ retransmissions) = \frac{\{\lambda[1+(k-1)(R+1)]T_s\}^l}{l!}\, e^{-\lambda[1+(k-1)(R+1)]T_s} \tag{5.29}$$

where $l = 0, 1, 2, \ldots$. The probability generating function for this number is

$$G(z) = e^{-\lambda[1+(k-1)(R+1)]T_s(1-z)} \tag{5.30}$$

Averaging over k, we have for the generating function of the number of

messages in the interdeparture interval

$$\begin{aligned} F(z) &= \sum_{k=1}^{\infty} G(z)(1-P)P^{k-1} \\ &= \frac{(1-P)e^{-\lambda T_s(1-z)}}{1-Pe^{-\lambda(R+1)T_s(1-z)}} \end{aligned} \tag{5.31}$$

By now the Markov chain is expressed in terms of data units. Under the assumption of compound Poisson arrival, the generating function of the number of data units in a message is $M(z)$. From this it follows that the probability generating function for the number of data units which arrive during the message service interval is

$$A(z) = \frac{(1-P)e^{-\lambda T_s[1-M(z)]}}{1-Pe^{-\lambda(R+1)T_s[1-M(z)]}} \tag{5.32}$$

Then we substitute equation (5.32) into equation (5.26) to find the probability generating function of the number of data units in the transmit buffer. We can find that

$$A'(1) = \rho + \frac{P\rho(R+1)}{1-P} \tag{5.33}$$

and

$$A''(1) = \frac{(\lambda T_s E[m^2] - \rho)(1+PR) + \rho^2[1+PR(R+2)]}{1-P} + \frac{2P\rho^2(R+1)(1+PR)}{(1-P)^2} \tag{5.34}$$

Thus, we can obtain queue length n and then from Little's formula the queueing delay is given by

$$D = n/\lambda$$

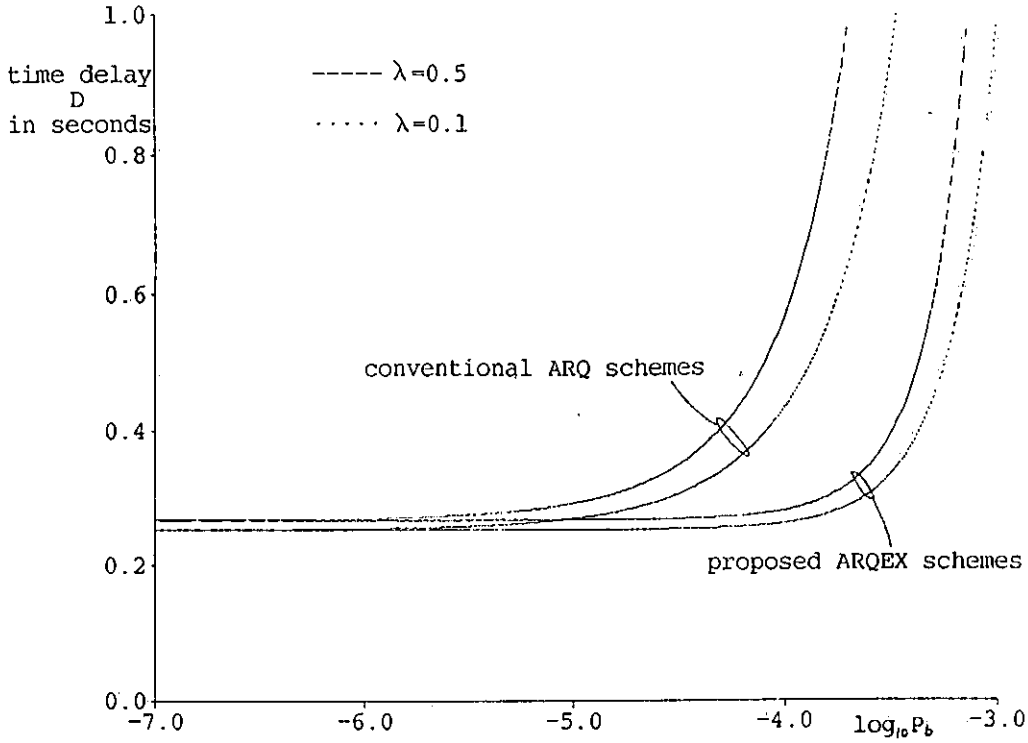


Figure 5.3: Relative Time Delay versus Bit Error Rate

$$= \frac{(1-\rho)\left(\rho + \frac{P\rho(R+1)}{1-P}\right)}{\lambda\left[1-\left(\rho + \frac{P\rho(R+1)}{1-P}\right)\right]} + \frac{A''(1)}{2\lambda\left[1-\left(\rho + \frac{P\rho(R+1)}{1-P}\right)\right]} \qquad (5.35)$$

where $A''(1)$ is given by equation (5.34).

Both Go-Back-N ARQ and ARQEX schemes have the same results given by equations (5.34) and (5.35). The difference between different schemes when using these equations lies in the meaning of the probability P. In the case of Go-Back-N, P means the probability of the event that one or more errors are detected in the received data unit. In ARQEX schemes, however, P denotes the probability of the event that more than k errors are detected in the received data unit so that retransmissions are required.

The relative comparisons of the numerical results based on equations (5.34), (5.35), (5.11), and (5.14), for the proposed ARQEX schemes and the conventional ARQ schemes, are shown in Figure 5.3. For simplicity but without missing the key thought, we have chosen the ARQEX in the special case 2 and the basic Go-Back-N ARQ as the representative cases

when plotting the curves. The average queueing delay is shown as a function of the bit error probability P_b with λ as a parameter. In computing these results the typical satellite channel was assumed to be the same as in the previous section when plotting throughput curves. Also it was assumed that the arrival process is simple Poisson so that each message or a data frame contains a single data unit. Obviously, the ARQEX schemes have less time delay than the basic Go-Back-N ARQ scheme. Again this is simply because, when errors occur, less data frames are required to be retransmitted in ARQEX than in Go-Back-N.

5.6 On Undetected Error Probability

The purpose of designing any ARQ schemes is that transmission errors caused by the channel noise can be controlled so that error-free data are delivered to the user. In the design of an ARQ scheme, it is often assumed that each transmission results in one of two outcomes, i.e., success or failure. Actually, three outcomes are possible: success, detected error, and undetected error. These three probabilities per transmission are denoted P_c, P_d, P_e. Obviously, $P_c + P_d + P_e = 1$.

A received frame is accepted by the receiver only if it either contains no errors or contains an undetectable error pattern. An undetectable error pattern can occur on the initial transmission of a frame or on any retransmission. The $P(E)$ that the received frame, in any existing ARQ system, is accepted and that an undetectable error pattern occurs in the frame is given by

$$\begin{aligned} P(E) &= P_e + P_d P_e + P_d^2 P_e + \ldots \\ &= P_e \left(\frac{1}{1 - P_d} \right) \end{aligned}$$

$$= \frac{P_e}{P_c + P_e} \tag{5.36}$$

The error-control code C_o can be properly chosen so that $P(E)$ is made very small [32]. We emphasize here that in the proposed ARQEX scheme, $P(E)$ may be eliminated in the case where all the transmitted data are from the predefined dictionary. This is another advantage of the ARQEX scheme.

In a conventional ARQ system, a high-rate error-detecting code, say an (n, k) linear block code, incorporated with a certain retransmission protocol, is used. When a message of k information bits is ready for transmission, $n - k$ parity-check bits are appended to it to form a codeword [31]. The codeword is then transmitted to the receiving end. When a frame is received, the receiver (or decoder) computes its syndrome which is a binary word computed by the decoder and used in making the decision as to which codeword was transmitted [7]. If the syndrome is zero, the received frame is a codeword in the code being used. In this case, the received frame is assumed to be error free and is delivered (with parity-check bits removed) to the user. If the syndrome of the received frame is not zero, the presence of errors is detected. In this case, the receiver discards the erroneously received frame and requests a retransmission of the same codeword via a feedback channel. When we implement a general ARQEX system, we may not use an (n, k) linear block code to do cyclic redundancy checks (CRC) in the same way of existing ARQ systems, because that the syndrome is not zero does not tell how many errors are detected in the received frame. We need to use an error-control code which can detect how many errors, say two or three, occurring in the received frame and detect burst errors at the same time.

Of course, in practice we may simply use an error correcting code instead of a spelling correcting system to correct the errors in the received messages.

Therefore, in the exact same way, the above method can also be used to analyze any Type-I hybrid ARQ/FEC scheme.

5.7 Summary

In this chapter, we have discussed performance issues of Type-I hybrid ARQ schemes by using the proposed ARQEX schemes as an example. The method used for performance analysis can be used to analyze any Type-I hybrid ARQ/FEC scheme.

The basic characteristic in the proposed ARQEX scheme is that the redundancy in real data is used for the purpose of correcting the errors in the received data. The quality of the transmission output may be further improved by correcting the undetectable errors in the conventional sense. Theoretically, the performance of ARQEX is much better than the basic GBN ARQ scheme, particularly for large-scale document transmission. Some interesting problems emerged from the ARQEX scheme should be studied further.

Chapter 6

Numerical Optimization of Type-I Hybrid ARQ with BCH Codes

6.1 Introduction

For a long time attention has focused on Type-II hybrid ARQ schemes. It has been recently recognized that for a given desired BER through a wide range, a Type-I hybrid ARQ scheme, such as one which has an appropriately designed t-error-correcting code, is best suited [42]. One of the most important problems in investigating Type-I hybrid schemes is how to choose a proper code since crossover points in signal strength exists among plain ARQ, Type-I and Type-II hybrid ARQ as far as efficiency is concerned, as shown in Figure 6.1.

Research related to find the optimum block length has already appeared in the literature [11]. In this chapter, the optimum code for error correction as well as optimum block length in Type-I hybrid ARQ schemes is investigated by a pragmatic optimization method. In Section 6.3, a new criterion called "overall throughput" is proposed to measure the performance capability in order to carry out optimizations. The optimum system parameters for Type-I hybrid ARQ with BCH error correcting codes are

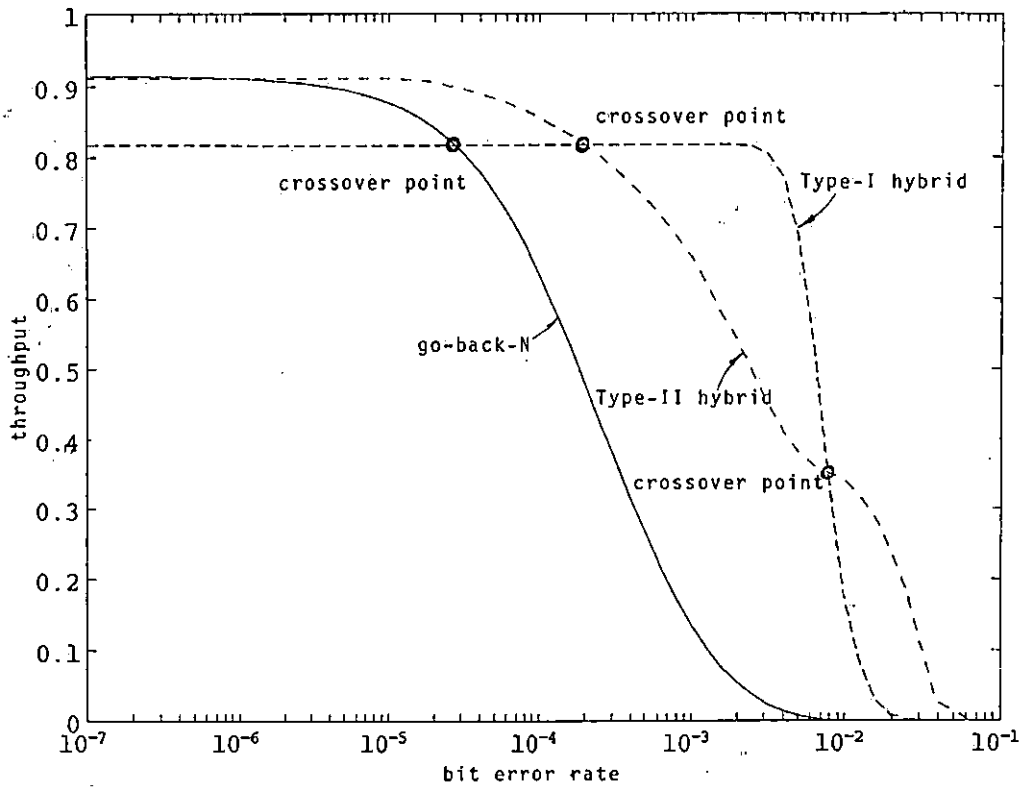


Figure 6.1: Throughput Performance of Hybrid ARQ Schemes

found numerically in Section 6.4. Finally, the complexity issue of system design is discussed for the optimum code. It is found that the optimum design offers the local maximum of overall throughput performance with the least system complexity.

6.2 Channel Statistics and Error Measurement

Two basic performance measures of a hybrid ARQ error-control system are reliability and throughput efficiency. Reliability is determined by the error detecting code used in the system. The optimum cyclic redundancy codes for error detection have been examined in the literature [37]. In this chapter we consider error correcting codes only. In order to carry out the system optimization, we need to form a composite performance criterion which gives the way in which each system parameter influences the whole.

Throughput efficiency provides such a possibility. But the throughput of ARQ schemes depends on the channel characteristics. As shown in Figure 6.1, the throughput of plain ARQ and Type-I hybrid ARQ is related to the channel bit error rate(BER). Without involving channel error statistics, we cannot tell whether Type-I hybrid ARQ is better than plain ARQ or not.

In practice, we must deal with data from the real world. The error statistics on many real channels are time-dependent. This leads to the design of adaptive error control systems [34]. But when real systems are measured under real operating conditions in the field, the errors are found to arrive mostly in bursts. There is the evidence which does not suggest that real systems are really well described by any model [8]. Many possible short burst periods may make adaptations difficult to carry out. Therefore, the overall system performance analysis has to include channel error measurement.

The parameter used to describe the performance of a digital system is the bit error probability, i.e. the incorrect reception of a single bit. Experimentally, the most often used parameter is the bit error ratio BER. This is defined as

$$BER = N_e/N_t = N_e/(r_b t_0) \tag{6.1}$$

where

N_e = number of bit errors in time interval t_0

N_t = number of transmitted bits in the time interval t_0

r_b = bit rate of a binary signal at the point where the measurement is performed

t_0 = measuring time interval or error counting time.

Where the error generation process is random and stationary and the errors are counted in a sufficiently long interval t_0, equation (6.1) can give an estimate of the error probability. The accuracy of the estimate increases as N_e increases. For a small error occurrence in the time interval t_0, i.e. for

small N_e, the measuring time interval t_0 becomes inordinately large if N_e is to be sufficiently large for reasonably good estimation. However, practical requirements on the measuring time interval usually limit the values which can be obtained for N_e. The minimum acceptable value of N_e seems to be about 10 and then the true error probability is contained in a range equal to ± 50 percent of N_e/N_t with a confidence coefficient of 90 percent [57].

For the purpose of system design and production testing, error performance measurement is tested under out-of-service conditions, i.e. a digital transmission link has been taken out of revenue or monitoring service. The test is done by inserting a test pattern into the input of the link, which simulates the normal traffic data stream. The output of the link is compared bit by bit with a locally generated error-free reference pattern in an error detector. The test pattern is usually a pseudo-random binary sequence which has a repetition period of $2^n - 1$ generated by a shift-register circuit. To have a reliable measurement on the inter-symbol interference effects, the register length n should be greater than, or equal to, the number of pulses which are affected by the channel response to a single pulse. As this requirement depends on characteristics of the transmission channel which are often unknown, a conservative value of n is usually chosen. To check the suitability of the test sequence, the error rates observed with different register lengths can be compared.

One method to obtain BER is using a fixed-length test sequence. Bit error rate measurement is performed according to the scheme in Figure 6.2. The generator produces a fixed-number test sequence of r bits. After transmission through the communication channel the test sequence is received at the receiver. Comparing the received sequence and the locally generated test sequence (which are synchronized) we find the number of errors k. The ratio k/r is the bit error rate which is shown on the display of the measuring device.

Consider the results of a BER measurement taking the following form:

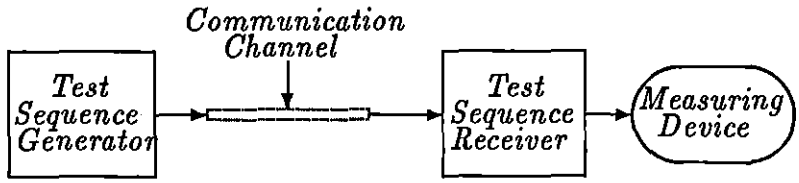


Figure 6.2: Block Diagram of Bit Error Rate Measurement

BER between 10^{-7} and 10^{-6} for x_1 percent of the time;

BER between 10^{-6} and 10^{-5} for x_2 percent of the time;

BER between 10^{-5} and 10^{-4} for x_3 percent of the time;

BER between 10^{-4} and 10^{-3} for x_4 percent of the time;

BER between 10^{-3} and 10^{-2} for x_5 percent of the time;

BER between 10^{-2} and 10^{-1} for x_6 percent of the time.

BER within other ranges for 0 percent of the time.

Clearly, we have $\sum_{i=1}^{6} x_i = 1$. One practical measurement with round-off discrete BER approximation may have the results shown as follows, which may reflect daily BER fluctuations of a Telex trunk [59].

$x(10^{-7}) = 0.03$; $x(10^{-6.5}) = 0.1$; $x(10^{-6}) = 0.4$; $x(10^{-5.5}) = 0.2$;

$x(10^{-5}) = 0.1$; $x(10^{-4.5}) = 0.07$; $x(10^{-4}) = 0.03$; $x(10^{-3.5}) = 0.01$;

$x(10^{-3}) = 0.04$; $x(10^{-2.5}) = 0.005$; $x(10^{-2}) = 0.01$; $x(10^{-1.5}) = 0.005$;

$x(other\ ranges) = 0$.

Here $x(10^{-7})$ is the percent time of BER between $10^{-7.25}$ and $10^{-6.75}$, and so on.

6.3 Performance Criteria for Optimization

Let us denote throughput as $T(P_b)$, then we introduce a new perfor-

mance criterion called *overall throughput* T_o defined as follows:

$$T_o = \int_0^1 x(P_b)T(P_b)\,dP_b \tag{6.2}$$

where P_b is the channel bit error probability, $x(P_b)$ is the bit error probability density function. In order to make the optimization numerically tractable, the following definition in discrete form is used:

$$T_o = \sum_i x(P_{bi})T(P_{bi}) \tag{6.3}$$

Obviously, the overall throughput provides a better description for evaluation of real ARQ error control systems compared with conventional definition of throughput efficiency which is a function of channel bit error rate. Based on the definition of overall throughput, the performance optimization problem of any type of ARQ scheme is of the form:

$$maximize \sum_i x(P_{bi})T(P_{bi}) \tag{6.4}$$

The objective of any optimization exercise is to select a vector of system parameters $Y = [y_1, y_2, ..., y_n]$ in such a way that some measure of system quality $Q(Y)$ is optimized. Overall throughput T_o provides such a proper measure of system quality for ARQ schemes. For a given channel BER measurement and a set of fixed system parameters, the overall throughput of the ARQ scheme is a real number between zero and one. In the following section, numerical optimization determines the optimum system parameters for any real channel condition which is obtained by the BER measuring method described above. We call it a pragmatic approach since the results are based on the practical BER measurement.

The performance of hybrid ARQ schemes as well as any of the possible variations is governed by the interrelationship between several key parameters, including error rate, data rate, channel propagation delay, block

length, required throughput efficiency, and error correcting capacity of the code used.

6.4 Optimum Design by Maximizing Overall Throughput

We denote the bit error rate as P_b, the data rate as C, the propagation delay as t_p, and the block length as $n + h$, where h is the block overhead used for the purpose of error-detection, block synchronization, and other necessary control functions; n is the length of the t-error-correcting inner code in a Type-I hybrid ARQ scheme. For given C, t_p, h as well as channel bit error rate measurement, the optimization problem is to find optimal values for n and t which provide maximum of the overall throughput with the least implementation complexity.

The inner code of the Type-I ARQ scheme uses a binary BCH code with the following parameters:

Code length:	$n = 2^m - 1$
Number of parity-check digits:	$n - k \leq mt$
Minimum distance:	$d_{min} \geq 2t + 1$

This code is capable of correcting any combination of t or fewer errors in a received codeword. The BCH codes are defined by the generator polynomial $g(X)$ which is specified in terms of its roots from the Galois field $GF(2^m)$ [7]. In order to carry out the numerical computation, we have to change the inequalities in the above definition to equalities. Fortunately, for systems with reasonable implementation complexity, we are restricted to small values of t (e.g., less than 10). By using computer search, it can be found that for $n = 511$ or greater, $n = 2^m - 1$ and $n - k = mt$ as long as $t \leq 16$. In practice, the maximum throughput normally occurs with n greater than 511.

The ARQ error control can be implemented using various protocols. If the GBN approach is used, the last N blocks in the buffer are retransmitted when one retransmission is requested. The throughput of GBN scheme can be found as follows:

$$T = \left(\frac{n}{n+h}\right) \frac{1-P_B}{1+(a-1)P_B} \tag{6.5}$$

where the block error probability P_B is

$$P_B = 1-(1-P_b)^{n+h} \tag{6.6}$$

and the parameter $a = 3 + 2t_pC/(n+h)$ as described in Chapter 5.

If a selective repeat scheme is used, only the block identified as being in error is retransmitted. The throughput is

$$T = \left(\frac{n}{n+h}\right)(1-P_B) \tag{6.7}$$

where P_B is same as above.

By using the above formula, it is easy to show that for Type-I hybrid go-back-N ARQ with BCH codes has throughput given by

$$T = \left(\frac{n-mt}{n+h}\right) \frac{1-P_B}{1+(a-1)P_B} \tag{6.8}$$

where a is as above and P_B is

$$P_B = 1-(1-P_b)^{n+h} - \sum_{i=1}^{t} \binom{n+h}{i} P_b^i (1-P_b)^{n+h-i} \tag{6.9}$$

If the selective repeat scheme is used in Type-I hybrid ARQ, then the throughput can be found as

$$T = \left(\frac{n-mt}{n+h}\right)(1-P_B) \tag{6.10}$$

where P_B is given by equation (6.9).

The numerical optimization is then carried out by

$$maximize \sum_i x(P_{bi})\, T(n, t, P_{bi} | \, given\, C, h, t_p) \qquad (6.11)$$

$$subject\ to: \quad n > 63 \ \ and \ \ t < 12$$

where the system constraints are determined by implementation complexity issues as well as throughput efficiency. From the engineering point of view, it is necessary to find the maximum overall throughput with both n and t as small as possible. With system constraints, only a local maximum may be found.

For the optimization problem addressed, it is difficult to use the well-known nonlinear optimization algorithms such as the *Fletcher-Reeves Method* [33], since line search steps may not be carried out in this case. Actually, based on the knowledge related to the problem, we find only limited computation is required to obtain the optimum result.

Our results of computation are based on the assumptions that the feedback channel is noiseless, a typical satellite link is used with propagation delay $t_p = 250msec$ and error statistics measurement results given in section 6.2, block overhead h is $48bits$, and the data rate C equals 4800bps. The computation is done on a UNIX workstation by using PROMATLAB package.

Part of the computation results is shown in Table 6.1 and Table 6.2. The contour plots of the overall throughput optimization are shown in Figure 6.3.

We see in the contour plot that there exists a local maximum point, ($n = 4095$, $t = 8$), so that the optimum block length of Type-I hybrid ARQ is about twice the length of plain ARQ [49]. In general, there may exist more local maximum points if we let n and t become large simultaneously without considering the implementation complexity. Therefore, what we

Computation Results of Overall Throughput
for GBN with t-error-correcting BCH Codes
(4800bps)

	t=1	2	3	4	5	6	7	8	9
n=63	0.5022	0.4534	0.4018	0.3492	*	*	*	*	*
127	0.6681	0.6359	0.5986	0.5609	0.5224	0.4831	0.4435	*	*
255	0.7897	0.7744	0.7513	0.7269	0.7024	0.6778	0.6527	0.6270	*
511	0.8625	0.8612	0.8505	0.8366	0.8219	0.8070	0.7922	0.7776	0.7630
1023	0.8984	0.9048	0.9054	0.9002	0.8928	0.8847	0.8763	0.8676	0.8589
2047	0.9117	0.9194	0.9251	0.9291	0.9294	0.9267	0.9228	0.9185	0.9141
4095	0.9124	0.9218	0.9257	0.9294	0.9338	0.9384	0.9416	0.9427	0.9420
8191	0.9033	0.9192	0.9242	0.9264	0.9280	0.9293	0.9311	0.9335	0.9368

* These are trivial cases where the throughput formula should be modified to obtain the correct data.

Table 6.1: Overall Throughput Computation 1

Computation Results of Overall Throughput
for S-R with t-error-correcting BCH Codes
(4800bps)

	t=1	2	3	4	5	6	7	8	9
n=63	0.5091	0.4567	0.4033	0.3496	*	*	*	*	*
127	0.6779	0.6408	0.6021	0.5627	0.5230	0.4833	0.4435	*	*
255	0.8022	0.7798	0.7558	0.7309	0.7054	0.6795	0.6534	0.6272	*
511	0.8783	0.8676	0.8540	0.8397	0.8254	0.8108	0.7958	0.7805	0.7649
1023	0.9160	0.9158	0.9104	0.9029	0.8947	0.8862	0.8778	0.8695	0.8613
2047	0.9263	0.9336	0.9364	0.9356	0.9327	0.9289	0.9246	0.9200	0.9153
4095	0.9233	0.9305	0.9364	0.9416	0.9452	0.9468	0.9467	0.9454	0.9435
8191	0.9175	0.9255	0.9286	0.9307	0.9333	0.9367	0.9406	0.9446	0.9481

* These are trivial cases where the throughput formula should be modified to obtain the correct data.

Table 6.2: Overall Throughput Computation 2

Computation Results of Overall Throughput
for GBN with t-error-correcting BCH Codes
(48kbps)

	t=2	3	4	5	6	7	8	9	10
n=63	0.4507	0.3997	0.3477	*	*	*	*	*	*
127	0.6313	0.5961	0.5581	0.5202	0.4822	0.4432	*	*	*
255	0.7662	0.7482	0.7245	0.6996	0.6746	0.6502	0.6257	*	*
511	0.8475	0.8455	0.8340	0.8200	0.8052	0.7900	0.7747	0.7597	0.7451
1023	0.8895	0.8946	0.8959	0.8905	0.8829	0.8749	0.8666	0.8580	0.8491
2047	0.9095	0.9133	0.9184	0.9233	0.9241	0.9213	0.9171	0.9126	0.9082
4095	0.9167	0.9207	0.9230	0.9257	0.9299	0.9349	0.9387	0.9400	0.9392
8191	0.9150	0.9218	0.9244	0.9261	0.9272	0.9281	0.9295	0.9318	0.9350

* These are trivial cases where the throughput formula should be modified to obtain the correct data.

Table 6.3: Overall Throughput Computation 3

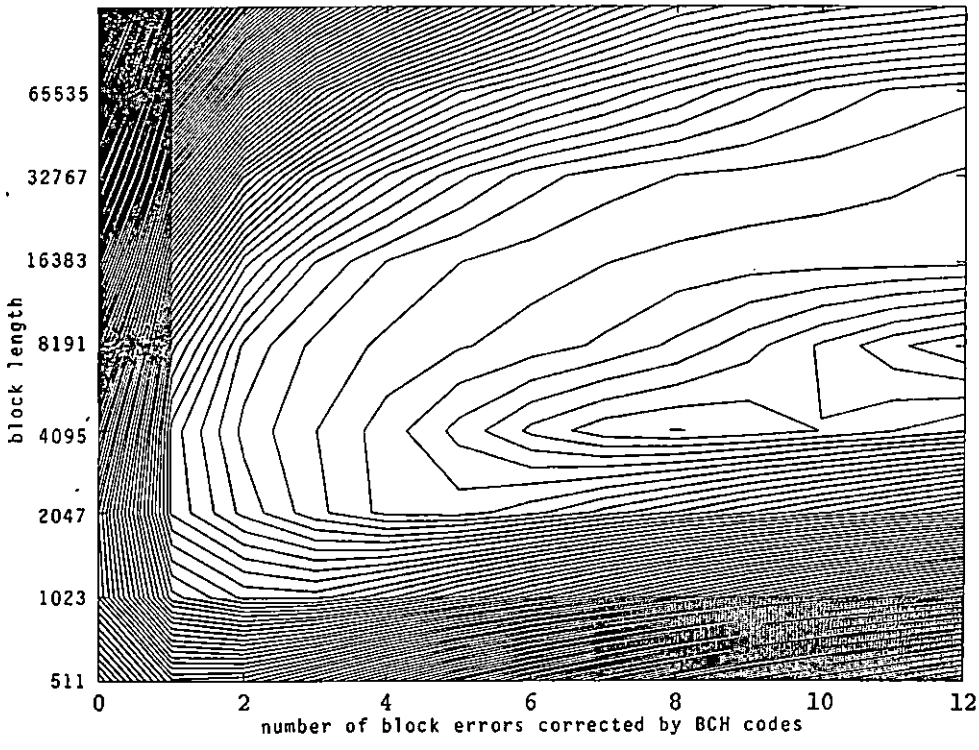


Figure 6.3: Contour Plot of Throughput for GBN with BCH Codes

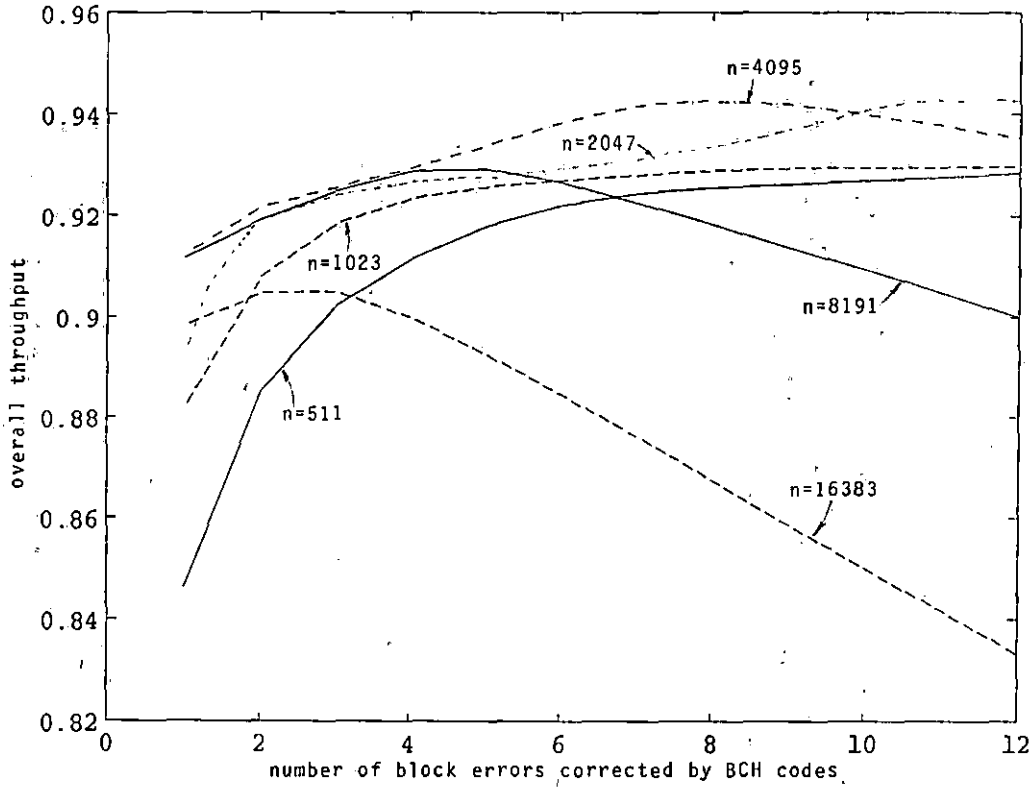


Figure 6.4: Search for the Optimum System Parameters

found by our contour plotting are those local maximum points with least system complexity. The improved overall throughput performances are shown in Figure 6.4 where we assume a data rate of $C = 4800$ *bps*. Table 6.3 contains the computation results for $C = 48$ *kbps*. Similar results can be obtained for other data rates.

6.5 Summary

The choice of error-control techniques and code parameters is primarily based on the channel characteristics. In this chapter, a pragmatic approach to design an optimum Type-I hybrid ARQ scheme with BCH error correcting code is proposed by introducing the *overall throughput* as a system performance measure, which is based on real channel measurement. The method can also be used to carry out numerical optimization of Type-II hybrid ARQ schemes. It has been shown that the proposed method has great significance in practical system design, since it provides the improve-

ment of overall throughput efficiency with least implementation complexity which is found to offer a local maximum in the throughput contour plot.

Chapter 7

Concluding Remarks and Future Work

7.1 Assessment of Approach Developed

In this work, a systematic approach to optimum coding design for Type-I Hybrid ARQ/FEC error control schemes has been developed.

Several simple, but fundamentally important ideas, concepts and methods are proposed to support our approach. They are summarized as follows:

First, a knowledge-combined expert system method is proposed for optimal coding selection. Two important issues for developing such a system have been identified. One concerns the construction of the knowledge base and the inference mechanism on a modified probabilistic inference network basis, and the other concerns the knowledge acquisition from sample or rule training using novel rule-transfer algorithms.

Second, a simple time delay analysis for Type-I hybrid ARQ schemes is carried out by using queueing theory. In this method, a Markov chain whose state is the number of messages in the system is embedded at the appropriate points. The solution to the steady-state equilibrium distribution of the chain allows us to find the average number of messages in the system. The application of Little's formula yields the average delay.

Finally, a pragmatic approach to design an optimum Type-I hybrid ARQ

scheme with BCH error correcting code is developed by introducing the overall throughput as a system performance measure, which is based on real channel measurement. This optimization provides the further improvement of overall throughput efficiency with reasonable implementation complexity which is found to offer a local maximum in the throughput contour plot.

7.2 Proposals for Future Work

This work has mainly dealt with basic issues for achieving optimum design of hybrid ARQ/FEC error control schemes. The proposed knowledge-combined coding selection system has not yet been fully implemented. To obtain an operational system, a complete knowledge base for the selection of error control coding must be developed. In addition, the proposed rule transfer algorithms for building and refining the coding selection knowledge base must be tested thoroughly to ensure that the system outputs agree with the decision of coding experts.

It is hoped that the practical system implementation in a well-supported software development environment will be carried out in the near future. Finally, the system should be extended to include a full range of error control coding techniques instead of Type-I hybrid ARQ schemes only.

Bibliography

[1] M. E. Anagnostou and E. N. Protonotarios, "Performance analysis of the selective repeat ARQ protocol," *IEEE Trans. Commun.*, vol. COM-34, pp. 127-135, Feb. 1986.

[2] B. G. Batchelor, *Practical Approaches to Pattern Classification*, Plenum Books, London, 1974.

[3] P. Benson, "Artificial intelligence assisted packet radio connectivity," *Electrical Communication*, vol. 60, no. 2, Nov. 1986.

[4] E. R. Berlekamp, R. E. Peile, and S. P. Pope, "The application of error control to communications," *IEEE Commun. Mag.*, vol. 25, pp. 44-57, Apr. 1987.

[5] D. Bertsekas, R. Gallager, *Data Networks*, Prentice-Hall, New Jersey, 1987.

[6] V. K. Bhargava, "Forward error correction schemes for digital communications," *IEEE Commun. Mag.*, vol. 21, pp. 11-19, Jan. 1983.

[7] V. K. Bhargava, D. Haccoun, P. Matyas, and P. Nuspl, *Digital Communication by Satellite*, Wiley, New York, 1981.

[8] M. B. Brilliant, "Observations of errors and error rates on T1 digital repeatered lines," *Bell Syst. Tech. J.*, vol. 57, pp. 711-746, Mar. 1978.

[9] H. Bruneel and M. Moeneclaey, "On the throughput performance of some continuous ARQ strategies with repeated transmissions," *IEEE Trans. Commun.*, vol. COM-34, pp. 244-248, Mar. 1986.

[10] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA, 1984.

[11] W. Bux, K. Kummerle, and H. L. Truong, "Balanced HDLC procedures: a performance analysis," *IEEE Trans. on Commun.*, vol. COM-28, pp. 1889-1898, Nov. 1980.

[12] P. R. Cohen, *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach*, Pitman Advanced Publishing Program, Boston, 1985.

[13] R. N. Cronk, P. H. Callaban, and L. Bernstein, "Rule-based expert systems for network management and operations: An introduction," *IEEE Network Magazine*, vol. 2, no. 5, pp. 7-21, Sept. 1988.

[14] J. Du, M. Kasahara, and T. Namekawa, "Separable codes on Type-II hybrid ARQ systems," *IEEE Trans. on Commun.*, vol. COM-36, pp. 1089-1097, Oct. 1988.

[15] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum Press, New York, 1988.

[16] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

[17] O. Ekeberg, "Robust dictionary lookup using associative networks," *Int. J. Man-Machine Studies*, vol. 28, no. 1, Jan. 1988.

[18] L. Earnest, "Can computers cope with human races?," *Communications of the ACM*, vol. 32, no. 2, Feb. 1989.

[19] R. Fikes and T. Kehler, "The role of frame-based representation in reasoning," *Communications of the ACM*, vol. 28, pp. 904-920, Sept. 1985.

[20] R. Forsyth and R. Rada, *Machine Learning: Applications in Expert Systems and Information Retrieval*, Ellis Horwood, Chichester, 1986.

[21] M. R. Genesereth, "The use of design descriptions in automated diagnosis," *Artif. Intell.*, vol. 24, pp. 411-436, Dec. 1984.

[22] A. Gupta and B. E. Prasad, (Eds.), *Principles of Expert Systems*, IEEE Press, pp. 211-214, 1988.

[23] J. F. Hayes, *Modeling and Analysis of Computer Communications Networks*, Plenum Press, New York, pp. 137-141, 1984.

[24] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, (Eds.), *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983.

[25] G. Jakobson, C. Lafond, E. Nyberg, and G. Piatetsky-Shapiro, "An intelligent database assistant with expert system capabilities," *IEEE Expert*, pp. 65-78, vol. 1, no. 2, Summer 1986.

[26] M. James, *Classification Algorithms*, Collins, London, 1985.

[27] T. P. Kehler and G. D. Clemenson, "An application development system for expert systems", *Systems and Software*, vol. 3, no. 1, pp. 212-224, Jan. 1984.

[28] L. Kleinrock, *Queueing Systems, vol. 1, Theory*, John Wiley & Sons, New York, pp. 191-194, 1975.

[29] J. Liebowitz, "If there is artificial intelligence, is there such thing as artificial stupidity?," *SIGART Newsletter*, no. 109, pp. 26-29, July 1989.

[30] J. Liebowitz ed., *Expert System Applications to Telecommunications*, John Wiley & Sons, New York, 1988.

[31] S. Lin and D. J. Costello,Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, New Jersey, pp. 85-116, 1983.

[32] S. Lin, D. J. Costello,Jr., and M. J. Miller, "Automatic-repeat-request error-control schemes," *IEEE Commun. Mag.*, vol. 22, pp. 5-17, Dec. 1984.

[33] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Addison-Wesley, Reading, MA, 1984.

[34] L. R. Lugand, D. J. Costello,Jr., and R. H. Deng, "Parity retransmission hybrid ARQ using rate 1/2 convolutional codes on a nonstationary channel," *IEEE Trans. on Commun.*, vol. COM-37, pp. 755-765, July 1989.

[35] K. Macleish, S. Theidke, and D. Vennergrund, "Expert systems in central office switch maintenance," *IEEE Commun. Mag.*, vol. 24, no. 9, Sept. 1986.

[36] L. Mantleman, "AI carves inroads: Network design, testing, and management," *Data Communications*, pp. 106-123, July 1986.

[37] P. Merkey and E. C. Posner, "Optimum cyclic redundancy codes for noisy channels," *IEEE Trans. on Inform. Theory*, vol. IT-30, pp. 865-867, Nov. 1984.

[38] A. M. Michelson and A. H. Levesque, *Error-control Techniques for Digital Communication*, Wiley, New York, 1985.

[39] M. L. Minsky and S. A. Papert, *Perceptrons*, MIT Press, 1969 and 1988.

[40] M. Moeneclaey and H. Bruneel, "Efficient ARQ schemes for high error rate channels," *Electron. Lett.*, vol. 20, pp. 986-987, Nov. 1984.

[41] J. N. Morris, "Another go-back-N ARQ under high error rate conditions," *IEEE Trans. Commun.*, vol. COM-26, pp. 187-189, Jan. 1978.

[42] M. Nakamura and T. Kodama, "Performance evaluation for ARQ schemes in power and/or bandwidth limited systems," *The Trans. of The IEICE*, vol. E72, pp. 494-501, May 1989.

[43] H. P. Nii, "Blackboard systems: The blackboard model of problem solving and the evolutions of blackboard architectures," *AI Magazine*, pp. 38-53, Summer 1986.

[44] J. P. Odenwalder, "Error control," *in* T. C. Bartee, (Ed.), *Data Communications, Networks, and Systems*, Chap. 10, H.W.Sams and Co., 1985.

[45] D. Prerau, "Selection of an appropriate domain for an expert system," *AI Magazine*, vol. 6, no. 2, pp. 26-30, Summer 1985.

[46] C. L. Ramsey, J. A. Reggia, D. S. Nau, and A. Ferrentino, "A comparative analysis of methods for expert systems," *Int. J. Man-Machine Studies*, vol. 24, pp. 475-499, May 1986.

[47] E. Y. Rocher and R. L. Pickholtz, "An analysis of the effectiveness of hybrid transmission schemes," *IBM J. Res. Dev.*, pp. 426-433, July 1970.

[48] A. R. K. Sastry, "Improving automatic-repeat-request (ARQ) performance on satellite channels under high error rate conditions," *IEEE Trans. Commun.*, vol. COM-23, pp. 436-439, Apr. 1975.

[49] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison-Wesley, Reading, MA, 1987.

[50] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, New Jersey, 1976.

[51] C. E. Shannon, "One man's approach to problem solving," *IEEE Commun. Mag.*, vol. 22, no. 5, pp. 123-126, May 1984.

[52] Special Issue, "Knowledge-based systems for communications," *IEEE J. on Selected Areas in Communications*, vol. 6, no. 5, June 1988.

[53] Special Issue, "Expert systems in network operations and management," *IEEE Network Magazine*, vol. 2, no. 5, Sept. 1988.

[54] Special Issue, "Artificial Intelligence in communications," *IEEE Commun. Mag.*, vol. 26, no. 3, Mar. 1988.

[55] S. L. Tanimoto, *The Elements of Artificial Intelligence*, Computer Science Press, 1987.

[56] M. Thandasseri, "Expert systems application for TXE4A exchanges," *Electrical Communication*, vol. 60, no. 2, 1986.

[57] A. A. R. Townsend, *Digital Line-of-sight Radio Links*, New York: Prentice Hall, New Jersey, 1988.

[58] D. Towsley and J. K. Wolf, "On the statistical analysis of queue lengths and waiting times for statistical multiplexers with ARQ retransmission," *IEEE Trans. Commun.*, vol. COM-27, pp. 693-703, Apr. 1979.

[59] D. A. Tugal and O. Tugal, *Data Transmission*, McGraw-Hill, New York, 1989.

[60] D. A. Waterman, *A Guide to Expert Systems*, Addison-Wesley, Reading, MA, 1986.

[61] W. W. Wu, D. Haccoun, R. E. Peile, and Y. Hirata, "Coding for satellite communication," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 724-748, May 1987.

[62] R. R. Yager, S. Ovchinnikov, R. M. Tong, and H. T. Nguyen, (Eds.), *Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh*, John Wiley & Sons, New York, 1987.

[63] Q. Yang and V. K. Bhargava, "A new ARQ scheme with error-correcting expert system and its performance analysis," *Proc. of IEEE Pacific Rim Conf.*, pp. 419-422, June 1989.

[64] Q. Yang and V. K. Bhargava, "Optimum Coding Design for Type-I Hybrid ARQ Error Control Schemes," *Electron. Lett.*, vol. 25, pp. 1595-1596, Nov. 1989.

[65] Q. Yang and V. K. Bhargava, "Building Expert Systems by a Modified Perceptron Network with Rule-Transfer Algorithms," *1990 IJCNN International Joint Conference on Neural Networks*, San Diego, California, June 1990.

[66] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338-353, 1965.

Glossary of Notation

Notation	*Definition*	*Typical Page Reference*
a	a is defined as $a = 1 + t_{out}/t_I$	40
ACK	Acknowledgment	40
ARQ	Automatic repeat request	2
$ARQEX$	A new ARQ scheme based on an expert system	36
$ATDM$	Asynchronous time-division multiplexing	49
BCH	BCH code is a class of linear block codes	63
BER	Channel bit error rate	59
C	Channel data rate	41
E_b/N_0	Ratio of energy per bit to noise power density	7
FEC	Forward error control	3
GBN	Go back N	44
$M(F)$	Membership number of a fuzzy set F	24
$M/G/1$	A queueing model with Poisson arrival, general service and a single server	50
$NACK$	Negative acknowledgment	40
$P(A)$	Probability of event A	22
$P(A\|B)$	Probability of event A conditioned on event B	22
$P(z)$	Probability generating function	50
P_b	Bit error probability	42
P_B	Block error probability	64
P_c	Success probability per transmission	54
P_d	Detected error probability	54
P_e	Undetected error probability	54
SAW	Stop and wait	44
SR	Selective repeat	45
t	Number of block errors corrected by BCH codes	63
T	Throughput	47
T_o	Overall throughput	62
t_I	Time required to transmit a frame	39
t_{out}	Timeout interval for retransmissions	39
$Type-I$	Type-I hybrid ARQ/FEC error control	34
$Type-II$	Type-II hybrid ARQ/FEC error control	34
$U(n)$	Unit step function	50

VITA

Surname: YANG *Given Name:* QING

Place of Birth: SHANGHAI, CHINA *Date of Birth:* Sept. 20, 1959

Educational Institutions Attended:

University of Science and Technology of China(USTC), Anhui, China. 1978 to 1982
University of Victoria, B.C. 1988 to 1990

Degrees Awarded:

B.S.E.E.(with distinction) USTC, China. 1982

Honors and Awards:

University of Victoria Fellowship, 1988 to 1990

Publications:

Yang, Q. and V. K. Bhargava, "A New ARQ Scheme with Error-Correcting Expert System and Its Performance Analysis," *IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, Victoria, B.C., pp. 419-422, June 1989.

Yang, Q. and V. K. Bhargava, "Optimum Coding Design for Type-I Hybrid ARQ Error Control Schemes," *Electronics Letters*, vol. 25, pp. 1595-1596, Nov. 1989.

Yang, Q. and V. K. Bhargava, "Building Expert Systems by a Modified Perceptron Network with Rule-Transfer Algorithms," *1990 IJCNN International Joint Conference on Neural Networks*, San Diego, California, June 1990.

Yang, Q. and V. K. Bhargava, "A Knowledge-Combined Approach to Optimum Selection of Error Control Coding," submitted to *IEEE Transactions on Aerospace and Electronic Systems*, 1990.

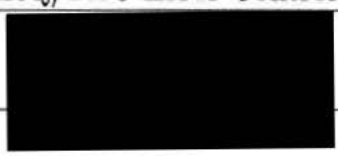
Yang, Q. and V. K. Bhargava, "Rapid Prototyping Knowledge-Based Systems by a Modified Neural Network," submitted to *IEEE Transactions on Knowledge and Data Engineering*, 1990.

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis (the title of which is shown below) to users of the University of Victoria Library, and to make *single copies only* for such users or in the response to a request from the library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my permission.

Title of Thesis

A Knowledge-Combined System Approach to Optimum Design of Type-I Hybrid ARQ/FEC Error Control Schemes

Author 

Qing Yang

April 15, 1990