1990

Usage-Dependent Information Systems Design.

Jin-soo Kim
Louisiana State University and Agricultural & Mechanical College

Order Number 9123208

Usage-dependent information systems design

Kim, Jin-Soo, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1990

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Usage-Dependent Information Systems Design

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Interdepartmental Program in Business Administration

by
Jin-Soo Kim
B.S., Yonsei University, 1982
M.B.A., University of Texas at Arlington, 1986
December 1990

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my major professor, Dr. Ye-Sho Chen, for his invaluable guidance, suggestions, and support throughout the period of this research.

I would also like to thank the members of my doctoral committee, Dr. James Pruett, Dr. Kwei Tang, Dr. Peter Kelle, and Dr. Bush Jones, for their constructive criticism and helpful suggestions. Especially, Dr. Peter Kelle's valuable suggestions made this research more worthwhile.

Special thanks are given to my mother and parents-in-law for their love and support. Finally, I am very thankful to my wife, Mi-Hyang Kim and my two children for their love, pray, understanding, and patience throughout my graduate studies in the United States.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF FIGURES

ABSTRACT

Usage-dependent phenomenon has been commonly observed in computer information systems (CIS). Since the performance of CIS greatly depends on the phenomenon, how to model it is an important CIS design issue. A usage process model (the Simon-Yule model) for modeling the usage-dependent phenomenon is proposed. The model is modified and successfully applied to the performance evaluation of self-organizing linear search heuristics. Analytical and empirical results indicate that the model provides a realistic performance evaluation of the heuristics and presents a solution to the research open problems which have been unsolved for more than two decades. Furthermore, the results lead to develop a self-organizing mechanism incorporating the usage process model for continuous speech recognition systems in the artificial intelligence arena.

CHAPTER 1

INTRODUCTION

Conventional wisdom has it that the more one uses something, the more the same thing will be used again. The phenomenon has commonly been observed in computer information systems (CIS). For example, a simple statement of the phenomenon is the 80-20 rule which states that 80% of computer usage involves only 20% of the resources (Heising 1953). It is logical to consider this usage-dependent phenomenon in information systems design in order to improve the usefulness and efficiency of CIS. When the information systems are defined as the usage-dependent information systems (UDIS), various applications can be found as a result of incorporating the usage-dependent phenomenon.

1.1 Usage-Dependent Information Systems

The literature indicates that UDIS have been proposed for use in various fields including telecommunications, computer science, and text modeling. Applications of UDIS are listed in Table 1.

Huffman Coding (1986) provides the most efficient and reliable representation of information storage and transmission in fields such as telecommunications and

Table 1: Applications of UDIS

Applications	References
Huffman coding	Hamming (1986)
Text compression	Salton (1989)
Usage-dependent menu design	Bayman et al. (1989)
Self-organizing files	Knuth (1973)
- VLSI circuit simulation program	Bently and McGeoch (1985)
- Interpreter design	Bitner (1979) Rivest (1976)
- Collision resolution in hashing files	Knuth (1973)
Index selection	Wiederhold (1987)
Type-token analysis of text generation	Wiederhold (1987)

computer science. In this type of code, characters are represented by a variable number of bits depending on the relative frequency of occurrence of the character. It is desirable to represent the most frequently occurring characters with the shorter bit patterns and the less frequently occurring ones with longer bit patterns.

A typical example is the Morse code, which uses the dot, dash, and space to express the letters of the alphabet or symbols. Sequences of codes are assigned to the letters of the alphabet based on the frequency of use for each letter; i.e., (.) for the highly frequent letter "E" and (-.-.) for the less frequently used letter "J". This technique significantly reduces the number of codes sent through transmission.

Another example is text (or data) compression. A study of composition text samples indicates the unevenness with which linguistic text elements occur. If the typical length of bits (i.e., 8- or 16-bit) is assigned to represent each word regardless of its frequency, a lot of processing time and storage space will be wasted. Huffman coding can be used to solve this problem. For example, the most frequently used character "0" is represented by a single-bit 0, and the least frequently used (%) is represented by a 16-bit pattern for a certain application.

Bayman et al. (1989) propose the usage-dependent menu design with Huffman coding. Most current menu systems employ static menu structures, requiring users to gain familiarity with large and complex fixed menu configurations. Against this, they suggest a dynamic menu system that continuously refines the menu tree based on users' search patterns so that users can access the information they need with few selections. It is obvious that usage-dependent menus would have an advantage over static ones during usage. Quattro, a spreadsheet software, allows users to change the menu structures to suit their preferences. Some users, however, might be uncomfortable with the dynamically changing, unstable menu structures. For this reason, it is essential to conduct studies of user preference to determine the utility of usage-dependent menu systems.

In a linear search list, initially unordered records are searched sequentially. In general, a record r_i will be accessed with probability p_i, where $p_1 + p_2 + ... + p_N = 1$. The time required to conduct a successful search is essentially proportional to the number of comparisons, C, which has the average value

$$C = p_1 + 2p_2 + ... + Np_N.$$

If we have the option of arranging the records in a linear

list in any desired order, then the above expression is minimized when

$$p_1 \geq p_2 \geq \cdots \geq p_N.$$

That is, by arranging the most frequently used records near the beginning, we can minimize the search time.

In most typical situations, however, the access probability p_i is not known a priori so that we cannot arrange the records by descending frequency of access (Knuth 1973, Bitner 1979). Several self-organizing linear search heuristics (Hester and Hirschberg 1986) are proposed to handle this problem. A list that is initially unordered can be dynamically reorganized according to usage patterns. By applying the heuristics, the most frequently used records will be moved to the front of the list and the less frequently used ones will move to the end as time goes by.

Wiederhold (1987) states that it is very important to choose the most selective index among various attributes for a query; i.e., choose the most frequently used attributes as the index. He also indicates the possible application of UDIS in type-token analyses of text generation, which are concerned with relationships between the number of different words (types) and the total number of words (tokens) in a literary text.

1.2 Problem Statement

As described above, UDIS have several advantages over information systems that do not consider the usage-dependent phenomenon; i.e., they save storage space, reduce the run time of programs, reduce communication time, and decrease search time. Because the performance of UDIS, however, depends on the distribution of usage patterns, one must determine the exact pattern in order to take full advantage of usage-dependent phenomenon and incorporate it effectively into systems design. For example, Bitner (1979) shows that, if the keys in data structures are optimally arranged according to the exact distribution of their usage patterns, substantial decreases in access time can result. Especially, if the distribution is in accordance with Zipf's law, the minimum cost is four or five times less than the random cost. Modeling exact usage patterns is essential, therefore, in designing UDIS. Two immediate questions follow: (1) how does one model the usage-dependent pattern? ; and (2) how does one take advantage of the patterns?

With regard to the first question, many approaches have been proposed during the last 50 to 60 years to describe the usage pattern. These include usage index, usage distribution, and usage process. Limitations inherent in the first two approaches, however, restrict their usefulness in real-world situations. The usage index approach shows

empirical results only, for example, without theoretical justification. The usage distribution approach provides the empirical distributions, but has two limitations: (1) estimating the parameters associated with the models is a very difficult task; and (2) the necessary statistical theory for making a rigorous test is not available (Chen 1988). To avoid these problems, the usage process approach is preferred. It provides a more constructive approach in modeling the usage-dependent phenomenon.

One way to answer the second question is to use the self-organizing linear search technique to take advantage of the usage-dependent phenomenon. Many self-organizing linear search heuristics have been proposed in order to dynamically rearrange the list according to usage patterns. This would involve moving more frequently accessed records near the front of the list and less frequently accessed ones near the end.

The literature shows that a lot of effort has been focused on performance evaluations of the heuristics to determine which is the most efficient under certain circumstances. The results of these evaluations provide valuable information to practitioners. For the purpose of evaluating the heuristics, three approaches are proposed: asymptotic, worst case, and amortized. These approaches

have provided many theoretical and empirical results of performance evaluation during the last two decades.

Despite the availability of evaluation results, several open problems remain (Rivest 1976, Hester and Hirschberg 1985). These are: (1) unrealistic assumptions of usage patterns; (2) the absence of theoretical bounds for the heuristics with reasonable assumptions; (3) the lack of a reasonable scheme for optimizing the heuristics; and (4) no good approach capturing the locality phenomenon in the request sequences. There is a strong need, therefore, for a more realistic performance evaluation approach, which would use reasonable assumptions of the usage pattern and at the same time solve the open problems previously identified.

1.3 Dissertation Purpose and Organization

The purpose of this dissertation is threefold. First, we propose to develop a usage process model that accurately describes the usage-dependent phenomenon, but is not subject to many limitations and is simple enough to use in the field. After a thorough literature review, we have chosen the Simon-Yule model of text generation (Simon 1955) as a promising approach. Second, we propose to apply the Simon-Yule model to the performance evaluation of the heuristics and, at the same time, solve the open problems described above. Third, a self-organizing mechanism for continuous

speech recognition is developed to show how the previous two findings can be incorporated into a UDIS design.

As a prelude to examining the research proposal, Chapter 2 discusses the various approaches to modeling usage patterns and their respective limitations. Also included in Chapter 2 is an overview of the Simon-Yule model which we propose to use in modeling the usage-dependent phenomenon. In Chapter 3, self-organizing linear search heuristics and the open problems previously cited are discussed. In Chapter 4, the analytical study of the performance evaluation of the heuristics using the Simon-Yule model are presented. In Chapter 5, the simulation results are presented. In Chapter 6, a self-organizing mechanism incorporating the usage process model for continuous speech recognition is proposed. Finally, the conclusion and further research issues are presented in Chapter 7.

CHAPTER 2

DESCRIPTION OF USAGE PATTERNS

There are three approaches to describing usage patterns: usage index, usage distribution, and usage process. We introduce these approaches in terms of text generation. The same concept can be applied to other applications without loss of generality.

2.1 The Usage Index Approach

The simplest way to describe a usage pattern is probably the usage index approach. Two usage indices are reviewed below: the 80/20 rule and the ABC analysis.

80/20 Rule

In 1953, Heising stated that 80% of a transaction deals with the most active 20% of a file and that the same rule applies to this 20%. More formally, the 80/20 rule can be described as

$$\frac{p_1 + p_2 + \cdots\cdots + p_{.20n}}{p_1 + p_2 + \cdots\cdots + p_n} \simeq .80 \text{ for all } n, \quad (2.1)$$

where p_i is the probability that a record i would be accessed. In terms of library management, the rule might state that approximately 80% of the circulations are accounted for by about 20% of the holdings.

ABC Analysis

In industrial engineering applications, an equivalent form of the 80/20 rule is the ABC analysis (Herron 1976), which ranks the items of a population in descending order of some activity, then develops the most appropriate techniques for handling the high-activity "A" group of items, the medium-activity "B" items, and the low-activity "C" items. The analysis is normally represented by the ABC curve, which is obtained by plotting the percent of ranked-to-total population against the corresponding cumulative percent of total activity value represented by that percent of the ranked population. In general, the ABC analysis shows that 20% of the ranked items account for 80% or more of the total activity, which is consistent with the 80/20 rule.

Drawbacks

The usage index approach is well accepted in the field because of its simplicity, but its lack of theoretical justification limits its application.

2.2 The Usage Distribution Approach

A more general description of the usage pattern is the usage distribution approach. Zipf's two laws of usage distribution are probably the most well known.

Zipf's First Law

In his book, *Human Behavior and the Principle of Least Effort*, Zipf (1949) stated that "if one takes the words making up an extended body of text and ranks them by frequency of occurrence, then the rank r multiplied by its frequency of occurrence, f(r), will be approximately constant." In symbolic form, this can be expressed as

$$f(r) = c/1,\ c/2,\ \ldots.,\ c/r$$
$$= cr^{-1},\quad r = 1,2,3,\ldots., \qquad (2.2)$$

where c is a positive constant. That is, if we plot f(r) vs. r on a log-log scale, then we see an approximately straight line with a slope of -1.

Zipf's first law attracted tremendous attention from many researchers and has been widely applied to many areas of computer science and text modeling, including: the applications listed in Table 1, program complexity in software engineering (Shooman 1983), key word distribution in bibliographic database design (Fedorowicz 1982a, 1982b), and information retrieval (Lancaster 1979). One of the

problems in the application of Zipf's law, however, is that its observation reveals only a crude approximation of the phenomenon; i.e., its simplicity cannot explain the concavity to the origin, as is usually the case with empirical log-log distributions. Several new formulations of Zipf's law, therefore, have been proposed. These new formulations strengthened the suitability of Zipf's law beyond the experimental evidence, which is strong in its own right. The most general formulation is perhaps the one proposed by Mandelbrot (1953):

$$f(r) = a(r + b)^{c}, \qquad r = 1,2,3,..., \qquad (2.3)$$

where $a > 0$, $c < 0$, and $b > -1$. The formulation is generally referred to as Mandelbrot's law of word frequency. Recent applications of this formulation include: secondary key indexing by Samson and Bendell (1985), and program complexity measures by Shooman (1983).

Zipf's Second Law

The study of Zipf's first law focuses mainly on words of high frequency. In contrast, Zipf's second law was motivated by two remarkable phenomena associated with words of low frequency of occurrence. If we observe and analyze the frequency of different words in long sequences of text and count f(n) as the number of words appearing n times,

then the ratio of the number of words occurring once (f(1)) to the number of different words in the text is approximately a constant 0.5. Also, the values of f(n)/f(1), n = 1,2,3,4,5, show an approximate pattern of 1, 0.33, 0.17, 0.10, and 0.07 (Booth 1967). Based on his first law, Zipf derived a formulation which he stated as the second law (1949). Booth (1967) argued that Zipf's formulation is only partially true and proposed a more general form as follows:

$$f(n) = a'(n^{c'} - (n+1)^{c'}),\ n = 1,2,3,\ldots, \qquad (2.4)$$

where $a' > 0$ and $c' < 0$. The formulation is referred to as Booth's law of word frequency (1967). Recent applications of this formulation include: indexed file performance evaluation by Fedorowicz (1987), and automatic text analysis by Pao (1978).

Drawbacks

A major difficulty in using Zipfian distribution is the estimation of the parameters associated with the formulation (e.g., a, b, c in equation (2.3)). To justify the estimated parameters, goodness-of-fit tests, which are a statistical test of a hypothesis that the sampled population is distributed in a specific way, are commonly used. There are several statistics used for a goodness-of-fit test. Among

those, the chi-square test is probably the most commonly used. The crucial assumption underlying the chi-square procedure is that the sample is randomly selected; i.e., the observations are independently and identically distributed. In practice, however, the observations may have substantial dependence, as when Zipfian data are collected as a time series; e.g., the data from a literary text.

As Ijiri and Simon (1977) pointed out, this is a questionable approach, because the necessary statistical theory for making a rigorous test is not available. Instead, they suggest that research should focus on the underlying mechanisms that can (1) explain the simplest form of Zipf's law as a first approximation, and (2) look for an additional mechanism that could be incorporated into the theory so as to lead to a better second approximation.

2.3 The Usage Process Approach

The most general approach to describe the usage pattern is the usage process, which provides the generating mechanism that are used to explain the phenomenon of Zipf's law. In this section, the performance of the three commonly used process models are discussed by examining their abilities to explaining Zipf's two laws.

The Multinomial Urn Model

In the multinomial urn model, the number s of an author's available vocabulary v_1, v_2, ...,v_s is assumed fixed, and each word, v_i, i = 1, 2, ..., s, is assumed to have a fixed probability p_i of being used each time the author writes a word, so that the probability of writing v_{i1} v_{i2} .... v_{in} is p_{i1} p_{i2} ... p_{in}, where $i_j \epsilon \{1, 2, .., s\}$, j = 1, 2, ..., n.

In general, since the occurrence of the next word strongly depends on the previous several words and the probability p_i is changing over the text generation, the model's assumptions (i.e., independence and fixed p_i) are unrealistic. In addition, Chen (1989) shows that the expected frequencies of words is inconsistent with equation (2.4); i.e., Zipf's second law. Thus, a dynamic probabilistic model, called Markov chain, was recognized as being a more realistic model for describing the usage pattern.

Markov Chain Model

In 1913 Markov showed that some verse by Pushkin, when reduced to a sequence of vowels and consonants, could be accurately represented as a first-order Markov chain. Let Y_1, Y_2, ..., Y_{k+1}, ..., Y_t, ..., be a k-th order Markov chain

and $Y_j = i_j$ if the word v_{ij}, $i_j \epsilon \{1, 2, \ldots, s\}$, is selected at the j's token of the text generation process. The Markov chain can be characterized by the initial probability:

$$P\ (Y_k = i_k,\ Y_{k-1} = i_{k-1},\ \ldots,\ Y_1 = i_1)$$

and by the following transition probabilities:

$$P_{ik+1} \mid i_{k-i}k-1,\ldots,i_1$$

$$= P\ (Y_{t+1} = i_{t+1} \mid Y_t = i_t\ ,\ldots.,\ Y_{t-k+1} = i_{t-k+1}),$$

$$= P\ (Y_{t+1} = i_{t+1} \mid Y_t = i_t\ ,\ldots.,\ Y_{t-k+1} = i_{t-k+1}),$$

$$Y_{t-k} = i_{t-k},\ \ldots.,\ Y2 = i_2\ ,\ldots.,\ Y_1 = i_1), \quad (2.5)$$

where $k = 1, 2, \ldots, t = k, k+1, \ldots.$ That is, the text generation process uses the word v_{ij}, $i_j \epsilon \{1,2,\ldots, s\}$, at the j's token. A succession of such models with increasing k can be regarded as a succession of approximations to strings of text. The first model is a stochastic process in which the words are selected according to a first-order Markov chain. This simple limitation makes it inadequate as a model of text generation since sentences exhibit constraints operating over much greater time spans. The second and succeeding models are stochastic processes in which the words are selected according to second and higher order Markov chains.

Among the several researchers sharing the idea of a Markov chain, Claude Shannon (1951) states that "The writing of English sentences can be thought of as a process of

choice: Choosing a first word from possible first words with various probabilities; then a second with probabilities depending on the first; etc. This kind of statistical process is called stochastic process, and information sources are thought of, in information theory, as stochastic processes." Nevertheless, there are critics opposed to using the use of the Markov chain as a model of text. For example, Miller and Chomsky (1963) showed that increasing the order of Markov chains does not, in the limit, yield an exact set of grammatically correct sentences. Chen (1989) also shows that the Markov chain model can hardly explain the Zipf's two laws.

The Simon-Yule Model

Simon (1955) proposed a more constructive approach which has the following steps:

(1) Begin with empirical data, not hypotheses.
(2) Draw a simple generation that approximately summarizes striking features of the data.
(3) Find limiting conditions under which deviations from a generalization are small.
(4) Construct simple mechanisms to explain the simple generalizations.

(5) Propose the explanatory theories that go beyond simple generalizations and make experiments for new empirical observations.

Based on his theory of modeling, Simon (1955) proposed the generating mechanism discussed below.

Basic Model

According to Simon (1955), the process of text generation can be described as a stochastic process. The stochastic process by which words are chosen to be included in written text is a twofold process. Words are selected by an author by processes of association (i.e., sampling earlier segments of his word sentences) and imitation (i.e., sampling from other works, by himself or other authors). Simon's selection processes are stated in the following assumptions, where f(n,t) is the number of different words that have occurred exactly n times in the first t words.

Assumption I: There is a constant probability, α, that the (t+1)-st word will be a new word-a word that has not occurred in the first t words.

Assumption II: The probability that the (t+1)-st word is a word that has appeared n times is

proportional to $n \cdot f(n,t)$-that is, to the total number of occurrences of all the words that have appeared exactly n times.

That is, assumption I and assumption II describe a stochastic process, in which the probability that a particular word will be the next one written depends on what words have been written previously. Based on the two assumptions, Simon derived

$$h(n) = \rho B(n, \rho+1), \quad n = 1,2,3,..., \tag{2.6}$$

where $h(n)$ is the expected relative frequency of words appearing n times, $\rho = \frac{1}{1 - \alpha}$ and $B(n, \rho+1)$ is the beta function with parameters n and $\rho+1$. Simon called equation (2.6) a Yule distribution because Yule's paper (1944), which predicted the modern theory of stochastic processes, derived the same equation in a study of a biological problem. Simon's approach is frequently cited as the Simon-Yule model of text generation.

From equation (2.6) and the index approach proposed by Chen and Leimkuhler (1989a), Chen (1989b) derived a more realistic formulation of equation (2.5), which provides a theoretical justification for the estimation and interpretation of the parameters associated with Zipf's first law.

Refinements of the Two Assumptions

Simon (1963) noted that the basic model is only a first approximation to the striking features of Zipfian data. He recommended further refinements of his model by modifying the assumptions so as to better represent the real world.

With regard to the first assumption of a constant probability α, a modification is introduced so that the entry rates for new words are a decreasing function of the length of the text. That is,

> Assumption I': There is a decreasing probability function $\alpha(t)$, $0 \leq \alpha(t) \leq 1$, that the $(t+1)$-st word be a new word - a word that has not occurred in the first t words.

Even with this slight modification, the problem is analytically difficult. Computer simulation methods, carried out by Simon and Van Wormer (1963), show a significant finding: the slight concavity toward the origin on a log-log plot is a result of a decreasing rate of entry of new words. In other words, the empirical data will continue to approximate the Simon-Yule distribution with a slight concavity to the origin, if the change in the rate of entry is not too rapid.

During 1964-1974, Simon successively modified Assumption II, in terms of business firms, to increase the realism of the model and show the effect of public policy on the size of firms. All the papers were collected in the monograph, *Skew Distribution and the Size of Business Firms* (1977). The modifications were based on empirical data and supported by economic theory. The two main refinements are: auto-correlated growth of firms, and mergers and acquisitions. They provide two different economic explanations for the concavity of the bilogarithmic firm-size distributions as observed in empirical data.

2.4 Summary

The several methods used to describe the usage pattern can be summarized as follows:

(1) The usage index approach is simple, but does not provide any theoretical justification.

(2) The usage distribution approach is more general than the usage index approach, but it has some difficulty estimating the parameters associated with the formulation and does not provide any rigorous statistical goodness-of-fit test to validate the distributions.

(3) The usage process models provide the most constructive approach in modeling the usage pattern. The multinomial urn model is simple, but its underlying assumptions--independence and fixed probability--are unrealistic and the result is inconsistent with Zipf's second law. The more dynamic and complicated model, the first-order Markov chain, was proposed, but its simple limitation makes it inadequate as a model of text generation since sentences exhibit constraints operating over much greater word spans. Furthermore, the Markov chain model involves too many parameters. This makes the model difficult to use to explain the simplicity of Zipf's law (Chen 1989).

(4) The Simon-Yule model of text generation is identified as the most promising approach for describing usage patterns. This is due to the following findings: (i) the model is simple, with only one parameter in the formulation; (ii) the model is constructive, providing a theoretical justification for the estimation of the parameter associated with Zipf's two laws; (iii) the model captures the dependent nature of record accesses in the request

search sequences, providing a realistic generating mechanism of text generation; (iv) the model is general, providing the explanation for such diverse phenomena as scientific publications, city sizes, income distribution, and biological species; and (v) the model is flexible, enabling successive refinements to increase realism.

CHAPTER 3

SELF-ORGANIZING LINEAR SEARCH

Linear search, a very simple way to retrieve data, has long been studied in the literature. In a linear search of a list of initially unordered records, the search progresses linearly; i.e., from the first record to the last until the requested record is found. In practice, it is seldom the case that all records are equally likely to be searched; some records will be accessed much more frequently than others. To take advantage of this usage-dependent phenomenon and enhance the performance of linear search, the order of the list must be dynamically changed so that frequently accessed records are moved to the front of the list, and less frequently accessed records are moved to the end.

During the last few decades, various self-organizing heuristics have been proposed to dynamically arrange the more frequently accessed records closer to the front of the list. Several measures have been developed to evaluate the relative performance of these heuristics.

In this chapter, we discuss applications of the self-organizing linear search, then review the details of all the

permutation heuristics, their performance evaluation, and open problems.

3.1 Applications

Bently and McGeoch (1985) justify the use of self-organizing linear search as follows:

(1) When n is small (i.e., at most, several dozen), the simplicity of the code can make it faster than more complex algorithms. This occurs, for example, when linked lists are used to resolve collisions in a hash table.

(2) When space is severely limited, sophisticated data structures may require too much storage space.

(3) If the performance of linear search is almost (but not quite) good enough, a self-organizing linear search list may give acceptable performance without adding more than a few lines of code.

Within these contexts, there are several applications of self-organizing linear search.

One common application is a list of identifiers maintained by a compiler or interpreter (Bitner 1979). The list cannot be initially ordered since frequencies are unknown, but, since most programs tend to access some identifiers much more often than others, the more frequently

accessed identifiers should be nearer the front of the search list.

One example is the list of identifiers maintained by a compiler or interpreter in the scatter table used by the LISP system at the University of California at Irvine (Hester and Hirschberg 1985). In this system, identifiers are hashed into a list of buckets, each of which is an unordered linear list of identifier descriptions. Virtually every command interpreted by the system involves one or more accesses to elements in the scatter table. Since most programs tend to access some identifiers more often than others, a lower average search cost can be obtained by moving the more frequently accessed identifiers to the front of the list.

Bently and McGeoch (1985) also describe that self-organizing linear search heuristics can improve the performance of a very large-scale integration (VLSI) circuit-simulation program that had two primary phases. The first phase read the description of the circuit from the symbol table, and the second phase simulated the circuit. The program spent five minutes in a set-up phase, most of which time was devoted to a linear search through a symbol table. Since this simulator was run on an on-line system, the five minute set-up phase was annoying to users. Incorporating a simple self-organizing search with about a

half-dozen additional lines of code reduced the set-up time to about 30 seconds.

3.2 Permutation Heuristics

The main feature that differs among heuristics is the moving distance of the accessed record. The heuristics move the accessed record forward by various distances according to their respective schemes, either constant or based on the location of the record or past events. The methods associated with these heuristics are explained using an initial linear list with four records: (A B C D).

Move-to-Front (MTF)

When the accessed record is found, it is moved to the front of the list, if it is not already there. All the records that the accessed record passes are moved back one to make room.

Transpose

In this heuristic, the accessed record, if not at the front of the list, is moved up one position by transposing it with the record just ahead of it.

Table 3.1: Example of MTF

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	**A** B C D
B	* *	**B** A C D
C	* * *	**C** B A D
D	* * * *	**D** C B A
A	* * * *	**A** D C B
B	* * * *	**B** A D C
C	* * * *	**C** B A D
D	* * * *	**D** C B A

COST = SUM OF PROBE TIMES/# OF ACCESS = 26/8 = 3.25

Table 3.2: Example of Transpose

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	**A** B C D
B	* *	**B** A C D
C	* * *	B **C** A D
D	* * * *	B C **D** A
A	* * * *	B C **A** D
B	*	**B** C A D
C	* *	**C** B A D
D	* * * *	**D** C B A

COST = 21/8 = 2.63

Count

When the record is accessed, the count of the accessed record is incremented and that record is moved forward to the first position in front of all records with lower counts. Thus, the list is always in decreasing order by the value of the counts of the records involved. This is the most accurate method; unfortunately, it requires substantial additional space for storage. Because of the additional space required, the count method has been considered in a different class from MTF and transpose, and has received less attention (Bently and McGeoch 1985) in the literature.

Table 3.3: Example of Count

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List	Count
A	*	**A** B C D	2 1 1 1
B	* *	A **B** C D	2 2 1 1
B	* *	**B** A C D	3 2 1 1
D	* * * *	B A **D** C	3 2 2 1
C	* * * *	B A D **C**	3 2 2 2
A	* *	B **A** D C	3 3 2 2
C	* * * *	B A **C** D	3 3 3 2
A	* *	**A** B C D	4 3 3 2

COST = 21/8 = 2.63

Move-Ahead-k

Rivest (1976) and Gonnet et al. (1979) proposed the move-ahead-k rule as a compromise between the relative extremes of the move-to-front and the transpose rules. It moves the record forward k positions, where k can be decided by users according to the distribution of accessed record order.

Table 3.4: Example of Move-Ahead-k (k = 2)

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	A B C D
B	* *	**B** A C D
C	* * *	**C** B A D
D	* * * *	C **D** B A
A	* * * *	C **A** D B
B	* * * *	C **B** A D
C	*	C B A D
D	* * * *	C **D** B A

COST = 23/8 = 2.88

Jump

Proposed by Hester and Hirschberg (1985), this rule uses a back pointer during the search. The pointer is later used as the destination for moving a record forward. The pointer is advanced to the probed record if and only if the probed record is not the accessed record. Jump can be a function of variables such as the current position of the back pointer, the position of the probed record, and/or the number of accessed records preceding the current one.

Table 3.5: Example of Jump

(Pointer Position: The Probed Record, and Suppose C in a Row)

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
		A B (C) D
D	* * * *	A B (C) D
C	* * *	A B C D
B	* *	A **C** B D
A	*	**C** A B D
D	* * * *	C A B D
C	*	C A B D
B	* * *	C A B D
A	* *	C A B D

COST = 20/8 = 2.5

Meta-Algorithm

The purpose of the meta-algorithm, applied in conjunction with the heuristic, is to slow the convergence of the latter by not moving records on the basis of single accesses only, thereby reducing the effects of a one-time access to a record. With regard to this method, four heuristics are suggested: (1) move-every-kth access; (2) k-in-a-row; (3) batched k; and (4) wait c and move.

Move-Every-kth Access

McCabe (1965) considers applying the permutation algorithm only once every k accesses to reduce the time spent reordering the list.

Table 3.6: Example of Move-Every kth Access

(Suppose k = 2 and Use MTF)

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	A B C D
B	* *	A B C D
C	* * *	**C** A B D
D	* * * *	C A B D
A	* *	**A** C B D
B	* * *	A C B D
C	* *	**C** A B D
D	* * * *	C A B D

COST = 20/8 = 2.5

k-in-a-row

This heuristic is applied only if the accessed record has been accessed k times in a row. If the record is accessed even twice in a row, the chances are greater that it will have additional accesses in the near future. This heuristic has the advantage of not requiring as much memory

as do count rules, since it is necessary to remember only the last record accessed and a single counter for the number of recent consecutive accesses to that record.

Table 3.7: Example of k-in-a row

(Suppose k = 3 and Use MTF; for k < 3, No Heuristic Is Used):

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	A B C D
B	* *	A B C D
C	* * *	A B C D
D	* * * *	A B C D
A	*	A B C D
B	* *	A B C D
C	* * *	A B C D
D	* * * *	A B C D
-> (The accessed record has been accessed twice in a row, so we apply the permutation algorithm - MTF.)		
C	* * * *	**C** A B D
	COST = 24/9 = 2.67	

Batched k

Gonnett et al. (1979) suggest the batched k heuristic with minor modification. This heuristic groups accesses

into batches of size k and applies the permutation algorithm only when all k accesses in a batch are to the same record. This tends to slow convergence down a bit more than k-in-a-row and provides a lower asymptotic cost.

Table 3.8: Example of Batched Heuristics

(Suppose k = 2 and Use MTF)

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	A B C D
D	* * * *	A B C D
B (Access same	* *	A B C D
B record here.)	* *	**B** A C D
C	* * *	B A C D
A	* *	B A C D
B	*	B A C D
D	* * * *	B A C D
C (" "	* * *	B A C D
C " ")	* * *	**C** B A D
COST = 25/10 = 2.5		

Wait C and Move

Bitner (1979) modified the k-in-a-row strategy and suggested the wait c and move heuristic, which incorporates

finite counters for each record. After a record has been accessed c times (not necessarily in a row), the heuristic is applied and the counter for that record is reset.

Table 3.9: Example of Wait C and Move

(Suppose c = 3; After That, the MTF Is Applied)

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List	Count
A	*	A B C D	1 0 0 0
B	* *	A B C D	1 1 0 0
B	* *	A B C D	1 2 0 0
D	* * * *	A B C D	1 2 0 1
C	* * *	A B C D	1 2 1 1
A	*	A B C D	2 2 1 1
B	* *	**B** A C D	3 2 1 1
--> Reset counter for B to 0.			
D	* * * *	B A C D	0 2 1 2
C	* * *	B A C D	0 2 2 2
COST = 22/9 = 2.44			

Hybrids

The MTF and transpose rules clearly have trade-offs concerning convergence and asymptotic cost. If it is known

in advance that the number of accesses will be small, move-to-front is probably the better heuristic. The transpose rule is better, however, if the number of accesses is expected to be large. A hybrid is a natural attempt to incorporate the best of both heuristics. Tenenbaum and Nemes (1982) suggest two classes of hybrids: POS(k) and SWITCH(k).

Hybrids - POS(k)

If the accessed record is found in a position <= k, it is transposed with its predecessor; otherwise, it is moved to the kth position, shifting all intervening records back one. Note that POS(1) is move-to-front, whereas POS(n - 1) is transpose.

Hybrid - SWITCH (k)

This method is the same as POS except that the use of move-to-front and transpose are reversed.

Table 3.10: Example of Hybrid - POS(k)

(Suppose k = 4 and Use MTF)

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	A B C D
B	* *	**B** A C D
C	* * *	B **C** A D
D	* * * *	B C **D** A
--> Use transpose method.		
A	* * * *	**A** B C D
B	* *	**B** A C D
C	* * *	**C** B A D
D	* * * *	**D** C B A

COST = 23/8 = 2.875

Table 3.11: Example of Hybrid - Switch (k)

(Suppose k = 4 and Use MTF)

Access Order	Probe Times 1 2 3 4	Self-Organizing Linear Search List
A	*	A B C D
B	* *	**B** A C D
C	* * *	**C** B A D
D	* * * *	**D** C B A
--> Use MTF method.		
A	* * * *	D C **A** B
B	* * * *	D C **B** A
C	* *	**C** D B A
D	* *	**D** C B A
A	* * * *	D C **A** B
B	* * * *	D C **B** A
C	* *	**C** D B A
D	* *	**D** C B A
--> Use transpose method.		

COST = 34/12 = 2.83

3.3 Performance Evaluation

For performance evaluation of the heuristics, several measurements have been suggested: asymptotic cost,

amortized cost, and rate of convergence (Hester and Hirschberg 1985). In general, the average search cost of a permutation heuristic for a given initial configuration of the list and a search sequence can be obtained by dividing the number of probes required to find the accessed record by the number of records accessed. The corresponding examples are given in the previous section.

A relative comparison can be obtained based on any heuristic, but costs of a heuristic are often compared with the cost of the optimal static ordering, in which the keys are initially arranged in decreasing order by their static probabilities and never reordered through the access sequence. Although the optimal static ordering is not optimal over all rules (because it is static rather than dynamic), it has been used as a basis for comparing the performance of the heuristics. Since the first three heuristics (i.e., MTF, transpose, and count rule) are representative of a large section, we will focus on these heuristics first and investigate the move-ahead-k heuristic in order to determine effects of a moving distance for an accessed record.

Asymptotic Cost

The asymptotic cost of a heuristic is the average cost for a single key over a search sequence and its initial

configuration of the list. As indicated above, the cost can be measured as the number of probes required to find the accessed records. A common assumption for analyses of asymptotic cost is that each record r_i independently accesses with a fixed probability p_i according to the probability distribution $P = \{p_1, p_2, \ldots p_i\}$.

Asymptotic cost $A_M(P)$ of the move-to-front (MTF) rule for probability distribution P has been derived by many researchers: McCabe (1965), Burville and Kingman (1973), Knuth (1973), Hendricks (1976), Rivest (1976), and Bitner (1979). The formula shows that $A_M(P)$ is, at most, twice the cost of the optimal static ordering, $A_o(P)$. However, Knuth (1973) shows that, under Zipf's distribution of search request, the MTF rule is approximately 1.386 times the optimal cost as $n \rightarrow \infty$.

Rivest (1976) also shows that the asymptotic cost of transpose A_T is less than or equal to that of $A_M(P)$ for every probability distribution. He also conjectured the transpose rule to be the optimal rule of all permutation rules.

Bitner (1979) shows that the search cost of the count rule is asymptotically equal to that of the optimal static ordering, and that the difference in cost between the two decreases exponentially with time, so that the count rule produces the ordering with the lowest expected cost for each

request.

In summary, for any probability distribution P, the previous asymptotic approach shows that

$$A_M(P) \leq 2A_O(P),$$

$$A_T(P) \leq A_M(P), \text{ and}$$

$$A_C(P) = A_O(P).$$

Worst-Case Cost

The worst-case cost of a permutation heuristic can be obtained by counting the worst-case number of comparisons over a given request sequence and an initial configuration, and multiplying that number by the number of accesses (i.e., the maximum value of the average search costs). By the given definition of cost, the worst case is bounded above by the total number of records in the list (i.e., n) since the cost is measured by the number of comparisons.

The worst-case cost of MTF has been shown by many (Bently and McGeoch (1985); Bitner (1979); Burville and Kingman (1973); Hendricks (1976); Knuth (1973); Rivest (1976); Sleator and Tarjan (1985)) to be no more than twice that of the optimal static ordering.

Amortized Cost

In many search processes, it is very rare that the worst case occurs at every step. Amortized analysis

considers this fact and combines aspects of worst-case and average-case analysis, and for many problems provides a measure of algorithmic efficiency that is more robust than average-case analysis and more realistic than wors-case analysis (Bently and McGeoch 1985, Sleator and Tarjan 1985). Compared to asymtotic cost or worst-case cost which determines a search cost for a single key, amortized analysis counts the worst-case number of comparisons made by a heuristic for any particular sequence of requests; because the cost is distributed over a series of requests.

Using an amortized argument, Bently and McGeosh (1985) show the different result from the asymptotic cost analysis; MTF and count rule are no more than twice that of optimal static ordering, but the worst-case performance of transpose could be far worse. They suggest that count and especially MTF rule are much more efficient than transpose. Sleator and Tarjan (1985) confirmed the results of Bently and McGeoch.

Rate of Convergence

In addition to the search costs, rate of convergence measures how quickly the permutation rules approach their steady states, where many further permutations are not expected to increase or decrease the expected search time significantly.

3.4 Open Problems

Despite the ample results of the previous performance analyses, there are several open problems which have remained unsolved for a long time. These are unrealistic assumptions, theoretical bounds, the optimization of heuristics, and locality.

Unrealistic Assumptions

A primary problem with most performance analyses is their unrealistic assumptions: incoming requests are independent of each other; and the probability of access for each record is equally fixed. It is well known (Hester and Hirschberg 1985) that incoming requests are dependent on the previous search performed and the access probability is being dynamically changed over the search sequence. Therefore, performance evaluation under more reasonable assumptions such that the search request is dependent on the previous search performed and the probability of access for a record is being changed over the search sequence would be valuable.

The Absence of Theoretical Bounds with Reasonable Assumptions

Most previous works have extensively analyzed the performance of MTF due to its wide acceptance from

practitioners, even though it is believed that transpose is theoretically better than MTF. The limitation of previous works again are the unrealistic assumptions such as the independence of the request sequence and the fixed probability of access to records.

The literature shows that the upper bound of MTF is asymptotically at most two times of that of the optimal static ordering ($AM \leq 2A_O$). Because of a dependency in the request sequence, the upper bound of MTF should be tighter than two times that of optimal static ordering.

Optimizing the Heuristics

Bitner (1979) shows that the MTF converges quickly and initially has a lower expected cost than transpose, but has a large asymptotic cost as the number of accesses increases.

The reason for this is obvious. In the initial random ordering list, many records with high access probability are far down in the list. These records should be moved to the front of the list to reduce the search cost. Obviously, MTF will perform well here since these records make large jumps and quickly rise to the front, thus converging quickly into its steady-state condition. Every time a record with a low access probability is accessed, however, it is moved to the front, thus increasing the cost of future accesses to many other records.

Rivest (1976) showed that transpose has a lower asymptotic cost than MTF. He also conjectured that this result extends to all the heuristics.

Intuitively, this conjecture is reasonable. After a long time, the frequently accessed records are near the front of the list, and the less frequently accessed records near the bottom. Occasionally, a low probability record will be accessed, and the MTF rule will move it all the way to the front on the basis of a single access. Unless that record is accessed again in the near future, its position will increase the expected search cost for other records since high probability records have moved down one position. The transpose avoids this potential error, and it is difficult for the less frequently accessed records to be advanced to the front. However, because of its conservative record movement, the rate of convergence is much slower than MTF.

A few hybrids were suggested to combine the best features of heuristics, such as using MTF initially and then switching to transpose; initially it converges quickly and maintains a low asymptotic cost. It is very difficult, however, to know the best switching time from MTF to transpose, and, as yet, no reasonable switching time has been suggested.

For example, Bitner (1979) proposed a hybrid rule that switches from MTF to transpose when the number of requests falls within a certain range. For Zipf's law he suggests switching from the MTF to the transpose rule when the number of requests is in the range of $\theta(N)$ to $\theta(N^2)$.

Since MTF performs better when the search request has a strong locality or unstable situation (i.e., large insertion rate), it is undesirable if the switching time can be determined only by a fixed number of accesses. When the number of requests falls within the range of switching time, for example, the list may not approach certain steady-state condition in which transpose tends to perform best. In addition to a fixed number of accesses, therefore, the steady-state condition of the list should be considered to determine a more accurate and realistic switching time.

Locality

It is well known (Rivest 1976, Hester and Hirschberg 1985, Bently and McGeoch 1985, Bellow 1987) that if the request sequence is a reference string generated from computer software, such a search sequence includes so-called "locality," where the relative frequency of sub-sequences of the request sequences may be significantly different from the overall relative frequencies. For example, we can consider the several occurrences of "integer" at the start

of a program, assignment of the form "x[i] := x[i] + 1", and "end;end;end" sequence.

The previous study (Bently and McGeoch (1985) and Bellows (1988)) shows that MTF performs better than transpose if there is a strong temporal locality in the request sequence. Bently and McGeoch (1985), however, just provide the empirical results without any model which can describe the locality. Bellows (1988) attempts to model the locality of the search sequence by applying the discrete auto-regressive process of order T or DAR (T), but he fails to relax the equal probability of access.

Since taking advantage of locality is one of the main reasons for using the heuristics in the first place, the performance analyses under the assumption of a strong locality in the request search sequence would be valuable. As yet, no good approach capturing the locality of accesses has been applied to the problem of measuring the performance of the heuristics (Hester and Hircheberg 1985).

CHAPTER 4

ANALYTICAL STUDY

The Simon-Yule model, which provides a promising approach to describing the usage pattern, is applied to the performance evaluation of self-organizing heuristics in this chapter. By applying the Simon-Yule model, the unrealistic assumptions identified in the previous chapter can be relaxed, allowing reasonable performance analyses of the heuristics. The theoretical bound of MTF is derived with realistic assumptions. Furthermore, the expected search cost of optimal static ordering is also derived for measuring the relative efficiency of MTF. Before discussing the analytical study, the major steps of the analysis are described.

4.1 Major Steps

Assume that N is the total number of record accesses, that t is the order of a record access, where t is $1 \leq t \leq N$ and f(n,t) is the number of distinct records that have occurred exactly n times in the first t record accesses.

The steps are mainly divided into two parts. The first two steps correspond, respectively, to Simon's two basic assumptions. The next two steps are added to incorporate the performance of self-organizing heuristics within the Simon-Yule approach.

[STEP 1]: Determine whether a new or an old record is accessed.

For each t ($1 \leq t \leq N$), a random number, a, is generated from the rectangular distribution with range $0 \leq a \leq 1$. If $a \leq \alpha(t)$, $f(1,t) = f(1,t-1) + 1$, where $f(1,0) = 0$; a new record is added to the end of the linear search list. Otherwise, GO TO [STEP 2].

[STEP 2]: Determine which group of old records is accessed.

A random number, b, is drawn from the rectangular distribution with range $1 \leq b \leq t$. Starting with $j = 1$, the cumulant of $j \cdot f(j,t-1)$ is computed, and compared with b until an n is found such that $\Sigma\ j \cdot f(j,t-1) \geq b$. Then $f(n,t-1)$ is decreased by 1, and $f(n+1, t-1)$ is increased by 1; $f(n,t) = f(n,t-1) - 1$; and $f(n+1,t) = f(n+1,t-1) + 1$. This is equivalent as the t-th record accesses the group of records that had previously occurred n times.

[STEP 3]: Find a specific record from the group of records chosen in STEP 2.

Create the cumulative distribution for the records within the group chosen in STEP-2, assuming that each record in the group will be equally accessed. A random number, c, is generated from the rectangular distribution with range $0 \leq c \leq 1$. Find a specific record by comparing c with the cumulative distribution.

[STEP 4]: Perform the heuristic.

Rearrange the linear search list according to each heuristic. If search sequences finish, stop. Otherwise, GO TO STEP 1.

4.2 A Theoretical Bound of MTF

As discussed in Chapter 3, one of the open problems is obtaining direct theoretical bounds on the behavior of the heuristics with reasonable assumptions. In this section, we derive the theoretical bound of the expected search cost of MTF based on the major steps described above.

Theorem 1: For a given t and α, the bound of the expected search cost of MTF, $E(C_M)$, under the Simon-Yule model of record accesses is

$$E(C_M) \leq 1/t\ [c_1\alpha^{t-1}+c_2\alpha^{t-2}(1-\alpha)+c_2\alpha^{t-3}(1-\alpha)^2,\ldots,$$
$$+\ c_2\alpha(1-\alpha)^{t-2}\ +\ c_t(1-\alpha)^{t-1}],\qquad (4.1)$$

where $c_1 = \dfrac{t(t+1)}{2}$,

$$c_2 = \frac{1}{2}\ (t-1)\ \{t^2 - \frac{(t-2)}{2}\},\ \text{and}$$

$$c_t = t.$$

Proof: To determine the expected search cost, the search trees are constructed for MTF according to the major steps. An example of the search trees is shown in Figure 4.1. Let α be an entry rate of a new record (Simon-Yule's Assumption I) and $(1 - \alpha)$ be an entry rate of an old record. Then $(1 - \alpha)\cdot \dfrac{nf(n,t)}{t}$ is the joint probability that the (t+1)-st record accesses to the group of records that has appeared n times (Simon-Yule's Assumption II). In addition, the access probability to a specific record in the group is

$$(1 - \alpha)\cdot \frac{nf(n,t)}{t} \cdot \frac{1}{f(n,t)} = (1 - \alpha)\frac{n}{t}\ .\qquad (4.2)$$

Based on the search trees, the expected search cost of MTF ($E(C_M)$) can be obtained as follows:

$t = 1,\ E(C_M) = 1$

$$t = 2,\ E(C_M) = \frac{1}{2}\{3\alpha + (1-\alpha)(1+1)\}$$

$$t = 3,\ E(C_M) = \frac{1}{3}\left[6\alpha^2 + \alpha(1-\alpha)\left\{\frac{(4+5)}{2} + \frac{4}{1}\right\} + 3(1-\alpha)^2\right]$$

$$t = 4,\ E(C_M) = \frac{1}{4}\left[10\alpha^3 + \alpha^2(1-\alpha)\left\{\frac{(7+8+9)}{3} + \frac{(7+8)}{2} + \frac{7}{1}\right\}\right.$$

$$\left. + \alpha(1-\alpha)^2\left\{\frac{(6+7)}{6} + \frac{(5+6+6)}{3} + \frac{10}{3} + \frac{5}{1}\right\} + 4(1-\alpha)^3\right]$$

$t = 5,$

$$E(C_M) = \frac{1}{5}\left[15\alpha^4\right.$$

$$+ \alpha^3(1-\alpha)\left\{\frac{(11+12+13+14)}{4} + \frac{(11+12+13)}{3} + \frac{(11+12)}{2} + \frac{11}{1}\right\}$$

$$+ \alpha^2(1-\alpha)^2\left\{\frac{(9+10+10+11+11+12)}{12} + \frac{(9+10+9+11)}{8}\right.$$

$$\left. + \frac{(8+9+10+9+9)}{6} + \frac{(8+8+8)}{4} + \frac{(8+9+8)}{3} + \frac{8}{3}(2) + 8\right\}$$

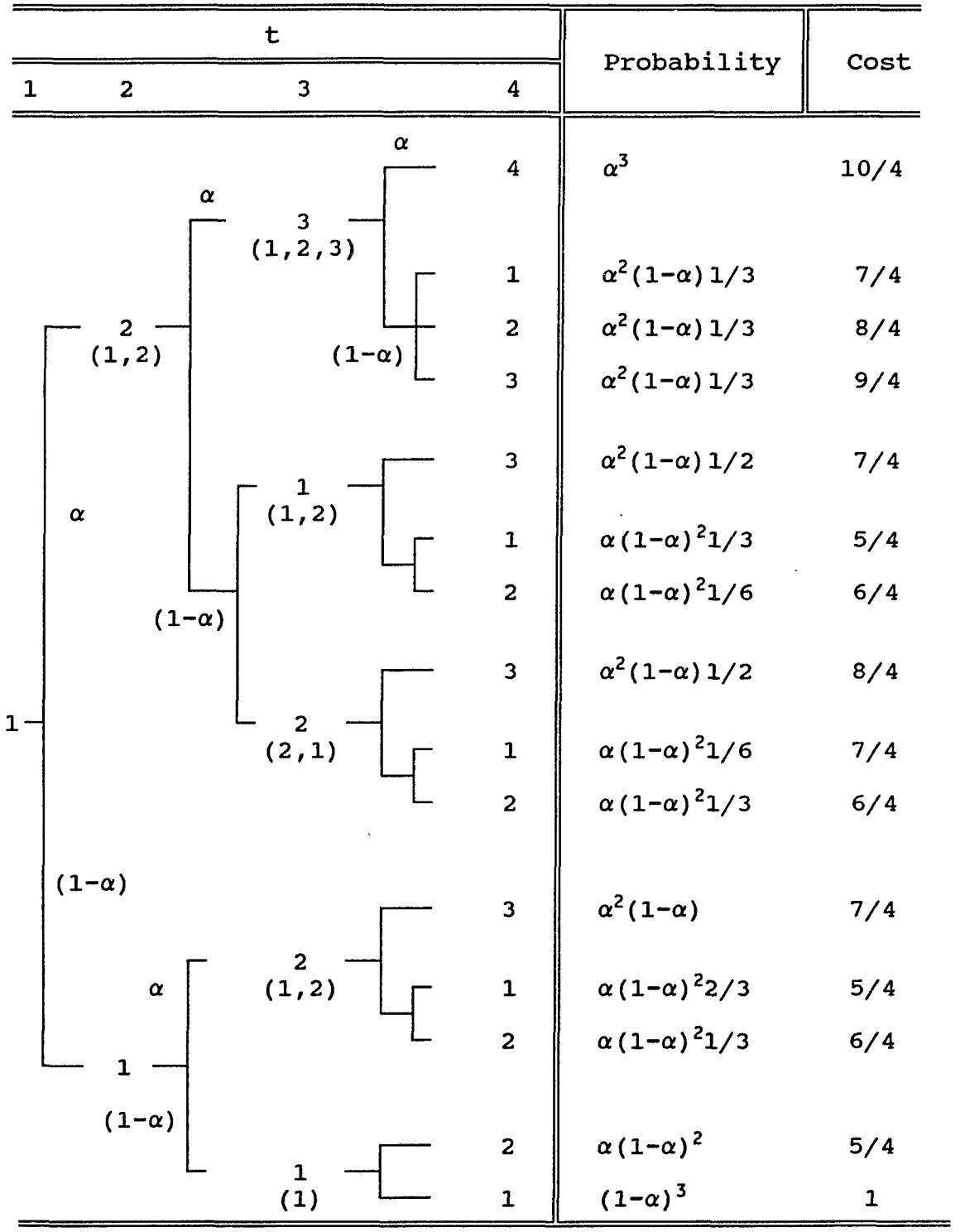
$$+ \alpha(1-\alpha)^3\left\{\frac{(7+7+8+8+9)}{12} + \frac{(6+7+7)}{4} + \frac{(7+7+8)}{6} + \frac{3}{4}(6)\right.$$

$$\left. + \frac{6}{2} + 1\right\} + \left.5(1-\alpha)^4\right]$$

. . .

. . .

Figure 4.1: Example of Search Tree of MTF



* ( ) indicates the list after permutation of MTF.

From these results, we derived the general form of the variables and the coefficients. The variable patterns are

$$t = 1, \quad 1$$

$$t = 2, \quad \alpha + (1-\alpha)$$

$$t = 3, \quad \alpha^2 + \alpha(1-\alpha) + (1-\alpha)^2$$

$$t = 4, \quad \alpha^3 + \alpha^2(1-\alpha) + \alpha(1-\alpha)^2 + (1-\alpha)^3$$

$$t = 5, \quad \alpha^4 + \alpha^3(1-\alpha) + \alpha^2(1-\alpha)^2 + \alpha(1-\alpha)^3 + (1-\alpha)^4$$

$$\vdots$$

$$t = k, \quad \alpha^{k-1} + \alpha^{k-2}(1-\alpha) + \alpha^{k-3}(1-\alpha)^2 + \ldots + \alpha(1-\alpha)^{k-2} + (1-\alpha)^{k-1}$$

When we define c_i as the coefficient of the ith variable, the coefficient patterns are derived through the following analyses:

i) $c_1 = 1, \quad 3, \quad 6, \quad 10, \quad 15, \quad 21, \ldots$

$\qquad\qquad \backslash/ \quad \backslash/ \quad \backslash/ \quad \backslash/ \quad \backslash/$

Diff. $= \quad 2 \quad 3 \quad 4 \quad 5 \quad 6, \ldots.$

$t = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6, \ldots$

$$\text{Therefore, } c_1 = \sum_{k=1}^{t} (1 + (k-1)) = t + \sum_{k=1}^{t} (k-1)$$

$$= t \left\{1 + \frac{(t-1)}{2}\right\}$$

$$= \frac{t(t+1)}{2} \qquad (4.3)$$

ii) $c_2 = \{ 1 \}$, $t = 1$

$= \{2/1\}$, $t = 2$

$= \{ \frac{(4+5)}{2} + \frac{4}{1} \}$, $t = 3$

$= \{ \frac{(7+8+9)}{3} + \frac{(7+8)}{2} + \frac{7}{1} \}$, $t = 4$

$= \{ \frac{(11+12+13+14)}{4} + \frac{(11+12+13)}{3} + \frac{(11+12)}{2} + \frac{11}{1} \}$,

$t = 5$

The general form of c_2 is derived by the following steps a) and b).

a) First element: 1, 2, 4, 7, 11, 16, ...

Diff. = 1 2 3 4 5, ...

Thus, the general form of the first element is:

$$a_1 = 1 + \sum_{k=1}^{(t-1)} k = 1 + \frac{(t-1)t}{2} \qquad (4.4)$$

b) Using equation (4.4) and t, we can further generalize the coefficient patterns. For example, for a given t,

$$c_2 = [\frac{\{a_1+(a_1+1)+(a_1+2) + ...+ (a_1+(t-2))\}}{t-1} + \frac{\{a_1+(a_1+1)+(a_1+2) + ...+ (a_1+(t-2))\}}{t-2} + ... + \frac{a_1+(a_1+1)}{t-(t-2)} + \frac{a_1}{t-(t-1)}]$$

$$= \left[\frac{(t-1)a_1 + (1+2 + \ldots + (t-2))}{t-1} \right.$$

$$+ \frac{(t-2)a_1 + (1+2 + \ldots + (t-3))}{t-2}$$

$$\left. + \ldots + \frac{(t-(t-2))a_1+1}{t-(t-2)} + \frac{(t-(t-1))a_1}{t-(t-1)} \right]$$

$$= \left[\frac{(t-1)(1+t(t-1)/2) + (1+2 + \ldots + (t-2))}{t-1} \right.$$

$$+ \frac{(t-2)(1+t(t-1)/2) + (1+2 + \ldots + (t-3))}{t-2}$$

$$+ \ldots + \frac{(t-(t-2))(1+t(t-1)/2) + 1}{t-(t-2)}$$

$$\left. + \frac{(t-(t-1))(1+t(t-1)/2)}{t-(t-1)} \right]$$

$$= \left[\frac{(t-1)(t(t-1)/2) + (1+2+\ldots+(t-2)+(t-1))}{t-1} \right.$$

$$+ \frac{(t-2)(t(t-1)/2) + (1+2+\ldots+(t-3)+(t-2))}{t-2}$$

$$+ \ldots + \frac{(t-(t-2))(t(t-1)/2) + (1+2)}{t-(t-2)}$$

$$\left. + \frac{(t-(t-1))(t(t-1)/2) + 1}{t-(t-1)} \right]$$

$$= \left[\frac{(t-1)\,\frac{t(t-1)}{2} + \sum_{k=1}^{t-1} k}{t-1} + \right.$$

$$+ \frac{(t-2)\,\frac{t(t-1)}{2} + \sum_{k=1}^{t-2} k}{t-2}$$

$$+ \ldots + \frac{(t-(t-2))\,\frac{t(t-1)}{2} + \sum_{k=1}^{t-(t-2)} k}{t-(t-2)}$$

$$\left. + \frac{(t-(t-1))\,\frac{t(t-1)}{t-(t-1)} + \sum_{k=1}^{t-(t-1)} k}{t-(t-1)} \right]$$

$$= \left[\frac{(t-1)\,\frac{t(t-1)}{2} + \frac{(t-1)t}{2}}{t-1} \right.$$

$$+ \frac{(t-2)\,\frac{t(t-1)}{2} + \frac{(t-2)(t-1)}{2}}{t-2}$$

$$+ \ldots + \frac{(t-(t-2))\,\frac{t(t-1)}{2} + \frac{(t-(t-2))(t-(t-3))}{2}}{t-(t-2)}$$

$$\left. + \frac{(t-(t-1))\,\frac{t(t-1)}{2} + \frac{(t-(t-1))(t-(t-2))}{2}}{t-(t-1)} \right]$$

$$= \left[\frac{(t-1)\left\{\frac{t(t-1)+t}{2}\right\}}{t-1} + \frac{(t-2)\left\{\frac{t(t-1)+(t-1)}{2}\right\}}{t-2} + \ldots + \frac{(t-(t-2))\left\{\frac{t(t-1)+(t-(t-3))}{2}\right\}}{t-(t-2)} + \frac{(t-(t-1))\left\{\frac{t(t-1)+(t-(t-2))}{2}\right\}}{t-(t-1)} \right]$$

$$= \frac{1}{2} \left[\{t(t-1)+t\} + \{t(t-1)+(t-1)\} + \ldots + \{t(t-1)+(t-(t-3))\} + \{t(t-1)+(t-(t-2))\} \right]$$

$$= \frac{1}{2} \left[(t-1)t(t-1)+t+(t-1)+(t-2)+ \ldots +(t-(t-3))+(t-(t-2)) \right]$$

$$= \frac{1}{2} \left[(t-1)^2 t + (t+t+\ldots+ t) - (0+1+2+ \ldots + (t-3)+(t-2)) \right]$$

$$= \frac{1}{2} \left[(t-1)^2 t + (t-1)t - \sum_{k=1}^{t-1} (k-1) \right]$$

$$= \frac{1}{2} \left[(t-1)^2 t + (t-1)t - \frac{(t-2)(t-1)}{2} \right]$$

$$= \frac{1}{2} (t-1) \left[(t-1)t + t - \frac{(t-2)}{2} \right]$$

$$= \frac{1}{2} (t-1) \left[t^2 - \frac{(t-2)}{2} \right] \qquad (4.5)$$

iii) There is no regular pattern for c_3 and c_4, but its value is less than c_2. In general cases, it is obvious that c_2 is larger than the following coefficients (i.e., c_3, $c_4,..,c_{t-1}$). In Figure 4.1, for example, c_2 is composed of all the worst cases (i.e., an insertion of a new record) for each search branch, and c_3 is composed of average cases (i.e., access to old records). Let us define c_{2i} to be the component of c_2 and c_{3i} to be that of the component of c_3 in ith search branch, respectively. Since c_2 consists of all the worst cases, each component of c_2 is larger than that of c_3, which is the case accessing to an old record. That is,

$$c_{21} > c_{31},\ c_{22} > c_{32},..,\ c_{2i} > c_{3i},\ ...,\ c_{2N} > c_{3N} \qquad (4.6)$$

$$= \sum_{i=1}^{N} c_{2i} > \sum_{i=1}^{N} c_{3i} \qquad (4.7)$$

$$= c_2 > c_3. \qquad (4.8)$$

When this concept is extended to more general cases, the following results are obtained.

$$c_2 > c_3 > c_4 >,....,\ > c_{t-1} \qquad (4.9)$$

Thus, based on the results of i), ii), and iii) we prove that for a given t and α,

$$E(C_M) = 1/t\ [c_1\alpha^{t-1} + c_2\alpha^{t-2}(1-\alpha) + c_3\alpha^{t-3}(1-\alpha)^2, .., + c_{t-1}\alpha(1-\alpha)^{t-2} + c_t(1-\alpha)^{t-1}]$$

$$\leq 1/t\ [c_1\alpha^{t-1} + c_2\alpha^{t-2}(1-\alpha) + c_2\alpha^{t-3}(1-\alpha)^2, .., + c_2\alpha(1-\alpha)^{t-2} + c_t(1-\alpha)^{t-1}], \qquad (4.10)$$

where $c_1 = \dfrac{t(t+1)}{2}$,

$c_2 = \dfrac{1}{2}(t-1)\left\{ t^2 - \dfrac{(t-2)}{2}\right\}$, and

$c_t = t$.

Table 4.1 shows that how much the bound is tight. The results show that the expected search cost from the bound (i.e., equation 4.10) is about 23 % increase over the $E(C_M)$ until t = 5. The bound seems to be tight enough for small number of accesses. Since the further analytical study is so difficult, a simulation study would be necessary to obtain more insight of the behavior of the bound. The simulation study to measuring the relative efficiency of the bound over the optimal static ordering is conducted and will be discussed in detail in Section 4.3.

Table 4.1: Comparisions Between the Bound and $E(C_M)$

t	$\alpha = 0.1$			$\alpha = 0.35$			$\alpha = 0.5$		
	Bound	E(MTF)	Bound/ E(MTF)	Bound	E(MTF)	Bound/ E(MTF)	Bound	E(MTF)	Bound/ E(MTF)
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.050	1.050	1.000	1.175	1.175	1.000	1.250	1.250	1.000
3	1.115	1.115	1.000	1.388	1.388	1.000	1.542	1.542	1.000
4	1.238	1.175	1.091	1.662	1.427	1.164	1.844	1.646	1.120
5	1.426	1.185	1.204	1.876	1.524	1.231	2.013	1.726	1.166

$E(MTF) = E(C_M)$

Corollary 1: At the initial stage (i.e., $t \leq 5$) the expected search cost of MTF is less than that of transpose.

Proof: Using the same procedure, we construct the search tree to calculate the expected search cost of transpose, $E(C_T)$. The result shows that until $t = 4$ and $E(C_M) = E(C_T)$. However, when $t = 5$,

$$E(C_T) = \frac{1}{5}\,[15\alpha^4$$

$$+ \alpha^3(1-\alpha)\,\{\frac{(11+12+13+14)}{4} + \frac{(11+12+13)}{3} + \frac{(11+12)}{2} + \frac{11}{1}\}$$

$$+ \alpha^2(1-\alpha)^2\,\{\frac{(9+10+10+11+10+12)}{12} + \frac{(9+10+9+11)}{8}$$

$$+ \frac{(8+9+11+9+9)}{6} + \frac{(8+8+8)}{4} + \frac{(8+9+8)}{3}$$

$$+ \frac{2}{3}\,(8) + 8\}$$

$$+ \alpha(1-\alpha)^3\,\{\frac{(7+7+8+8+9)}{12} + \frac{(6+7+7)}{4} + \frac{(7+7+8)}{6} + \frac{3}{4}(6)$$

$$+ \frac{6}{2} + 1\} + 5(1-\alpha)^4]$$

$$= E(C_M) + \alpha^2(1-\alpha)^2\,\frac{1}{12} \qquad (4.10)$$

$$\text{Since } \alpha^2(1-\alpha)^2\,\frac{1}{12} > 0, \quad E(C_M) \leq E(C_T). \qquad (4.11)$$

However, we found that the other coefficients, c_3, c_4,....., c_{t-1} are having differences as the number of accesses increases. Since the literature shows (Bitner 1979) that transpose is superior to MTF when the list reaches its steady-state where many further permutation are not expected to increase or decrease the expected search time significantly (i.e., the number of accesses increases), we may conjecture that the upper bound of transpose would be less than that of MTF as the number of accesses increases. However, further analytical study becomes intractable since the number of possible occurrences to be considered increases by approximately 2^t as t increases. Because of this, it is necessary to use the simulation method to obtain insights into the relative performance of different heuristics as the model parameters vary.

4.3 Relative Efficiency of MTF

The literature shows that the relative efficiency of heuristics is compared with the expected search cost of optimal static ordering. Most previous approaches, assuming independence of search sequences and equal access probability, conclude that the average search cost of MTF is at most two times that of optimal static ordering; i.e., $A_M \leq 2A_0$. Since the Simon-Yule model relaxes the previous unrealistic assumptions, the theoretical bound of the expected search cost of MTF should be less than two times

that of optimal static ordering. For measuring the relative efficiency of MTF, therefore, the expected search cost of optimal static ordering must be derived.

The Expected Search Cost of Optimal Static Ordering

A sequential search (Knuth 1973, Bitner 1979, Bently and McGeoch 1985) begins at the beginning of a file and goes through each record until a desired record is found or the end of the file is reached. Suppose there are N records in the file, and let p_i be the probability that record k_i will occur, then

$$\sum_{i=1}^{N} p_i + q = 1, \qquad (4.12)$$

where q is the probability that the record is not in the file. Supposing the file is huge, we can reasonably assume that $q = 0$, then $\sum_{i=1}^{N} p_i = 1$. Let $E(C_0)$ be the expected number of comparisons to search a record, then $E(C_0) = \sum_{i=1}^{N} ip_i$. If we have an option to arrange the records in any order we desire, then $E(C_0)$ reaches its minimum if

$$p_1 \geq p_2 \geq .... \geq p_N. \qquad (4.13)$$

In other words, the records are arranged by descending frequency of access, such that the most frequently used records appear close to the beginning.

Chen (1990) derived a more realistic formulation of Mendelbrot's law of word frequency from the Simon-Yule

distribution. That is, the three parameters of equation (2.3) can be estimated as follows:

$a = (v\rho\Gamma(\rho))^{1/\rho}$, where $\rho = \frac{1}{(1-\alpha)}$ and v is the total number of distinct words.

$b = \frac{1}{m} \Sigma\ (-\epsilon(n_i))$, when $\epsilon(n_i)$, $1 \le i \le m$ are approximately equal, and

$c = -1/\rho$.

If the two underlying assumptions of the Simon-Yule model hold and the error terms $\epsilon(n_i)$, $i = 1, 2, 3, \ldots, m$, are approximately equal, then an appropriate distribution for p_i $i = 1, 2, 3, \ldots, N$ would be

$$p_i = a(i+b)^c,\ i = 1, 2, \ldots, N, \qquad (4.14)$$

where $a = (v\rho\Gamma(\rho))^{1/\rho}$, $b = \frac{1}{m} \Sigma\ (-\epsilon(n_i))$, and $c = -1/\rho$.

Note that equation (4.14) is equivalent to the more realistic formulation of Mendelbrot's law of word frequency which are derived by Chen (1990). When $b = 0$ and $c = -1-\theta$, for a small positive value of θ, the equation (4.14) reduces to Schwartz's proposal for Zipf's law. Thus, the expected search cost of optimal static ordering is

$$E(C_0) = \sum_{i=1}^{N} ai(i+b)^c. \qquad (4.15)$$

Measuring The Relative Efficiency of MTF

The relative efficiency of MTF over the optimal static ordering can be measured by $E(C_M)/E(C_0)$. However, it is observed that it is hard to find a closed-form for the ratio of $E(C_M)/E(C_0)$. The difficulty comes from the fact that it involves so many parameters which cannot be reduced. This leads to the simulation method for measuring the relative efficiency of MTF.

The simulation study based on the equation (4.1) and (4.15) was conducted with various entry rates ranging from α = 0.1 to 0.9 until the total number of accesses is 500. The results shown in Table 4.2 indicates that the expected search cost of MTF is at most 1.58 times of that of the optimal static ordering. The upper bound of 1.58 is significantly lower than two proposed by most previous approaches, which employ the unrealistic assumptions such as independence and fixed probability of request accesses. Note that as α increases, the relative efficiency of MTF is also improved.

Another important characteristic of the bound of MTF would be that the vaules of the bound depend on the number of accesses and the entry rates. Figures 4.2 and 4.3 indicate how the bound is changed for different α and t values. Figure 4.2 shows that as α increases with the fixed number of accesses (i.e., t = 500), the bound also increases

more rapidly. On the other hand, Figure 4.3 shows that as the number of accesses increases, the bound increases very rapidly for the large values of α (i.e., α = 0.35 and 0.5), but it increases gradually for the small vaule of α (i.e., α = 0.1).

Table 4.2: Relative Efficiency of a Bound of MTF

The Values of Alpha	$E(C_0)$	Bound	Bound/$E(C_0)$
0.10	7.21	11.37	1.58
0.18	15.43	23.61	1.53
0.25	25.01	35.97	1.43
0.35	48.54	69.02	1.42
0.40	63.29	90.96	1.43
0.50	82.15	115.66	1.41
0.70	116.22	162.71	1.40
0.90	166.16	229.30	1.38

$E(C_0)$: The average search cost of optimal static ordering

Bound: The bound of MTF

4.4 Findings of Analytical Study

Using the Simon-Yule model, the previous unrealistic assumptions are relaxed, providing reasonable performance

analyses. In order to solve one of the open problems as well as to provide realistic comparisons between the heuristics, the theoretical bound of MTF is derived based on the search tree incorporating the Simon-Yule model and the corresponding heuristic. For measuring the relative efficiency of MTF, the expected cost of optimal static ordering is also derived. The results show that the bound of the expected search cost of MTF is at most 1.58 times that of optimal static ordering. This bound is significantly lower than two times proposed by previous approaches, which assume that request sequences are independent and the accessed probabilities are fixed. It is also interesting that as α increases, the relative efficiency of MTF over optimal static ordering improves. This observation clearly supports the corollary 1; i.e., at an initial stage, MTF performs better.

Despite the theoretical bound of MTF with reasonable assumptions, there is room to do more complete analyses. First, a tighter bound of MTF would be derived. Second, a bound of transpose which would be different from that of MTF should be derived to allow more realistic comparison. Third, a closed-form for the ratio between MTF and optimal static ordering would be desirable.

Figure 4.2: The Bound of MTF
According to the Values of Alpha

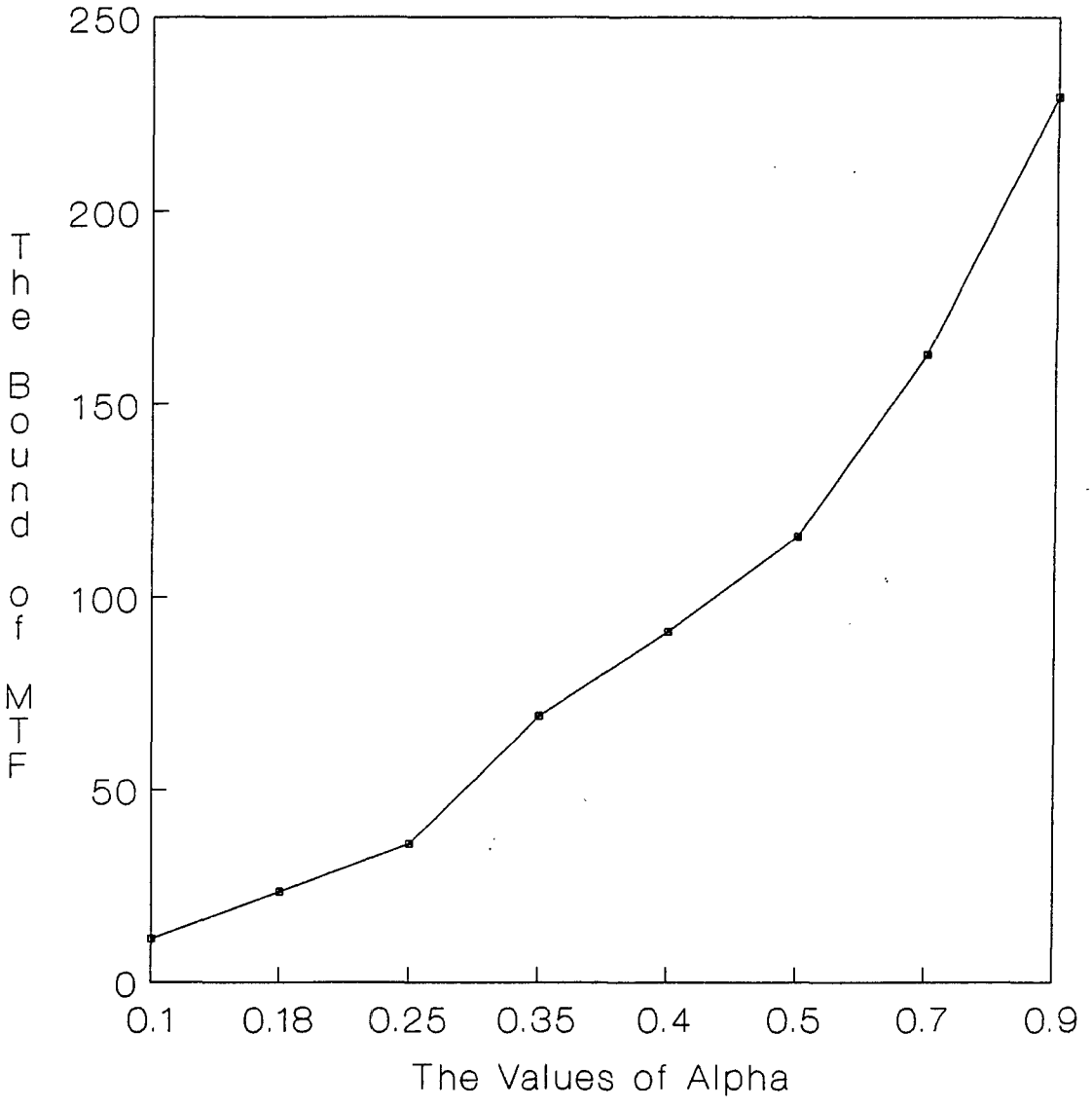
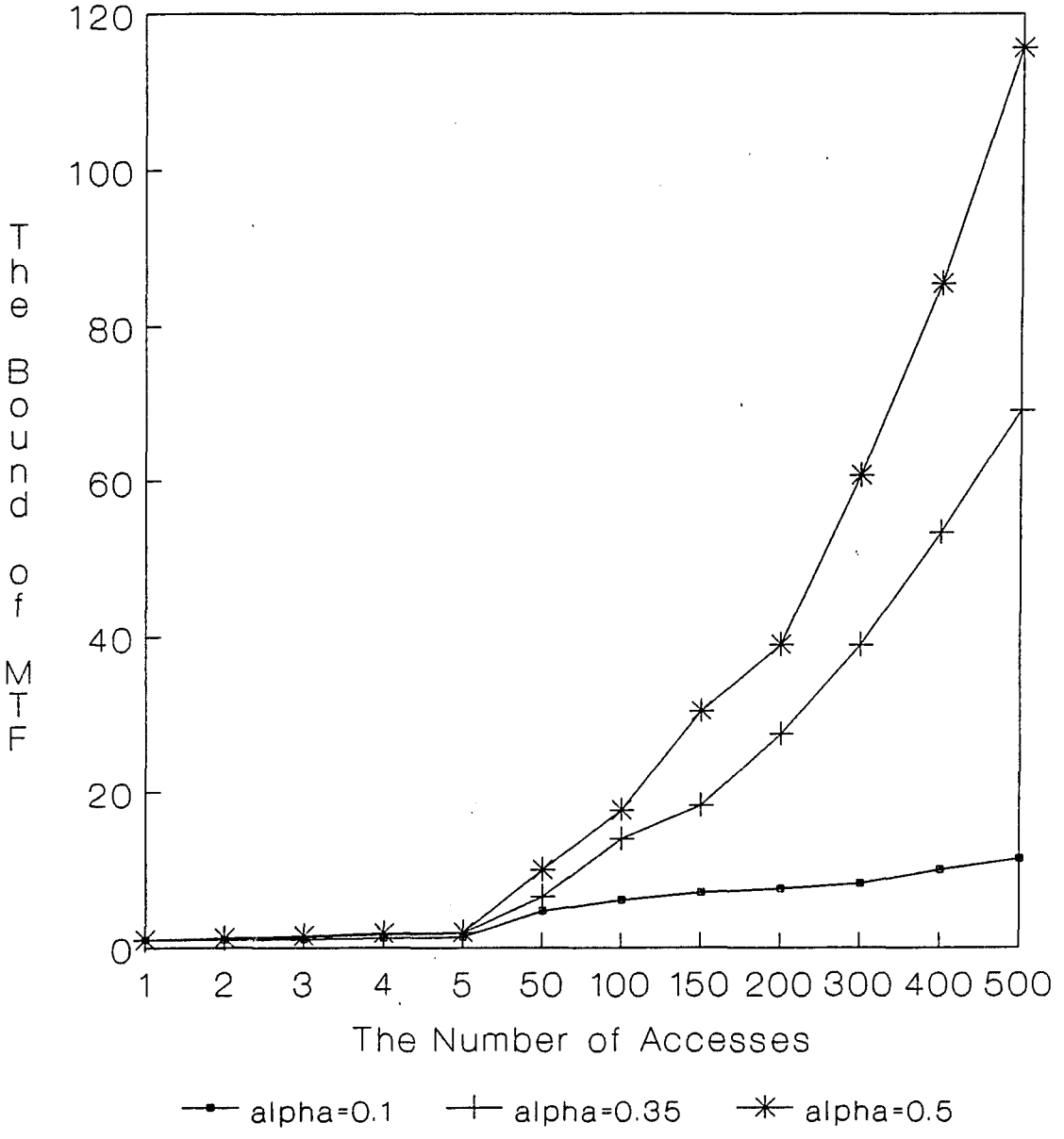


Figure 4.3: The Bound of MTF According to the Number of Accesses



CHAPTER 5

SIMULATION RESULTS

In this chapter, simulation analyses are performed to further study the behavior of the heuristics as well as to solve the open problems, which are identified in Chapter 3. We first focus on the three widely used heuristics: MTF, transpose, and count followed by further analyses on move-ahead-k. The chapter is divided into five sections. In section 5.1, results with constant entry rates (Assumption I) ranging from 0.1 to 0.5 are presented. In section 5.2, results with decreasing entry rates (Assumption I') are presented. In section 5.3, a proposed hybrid rule which is developed by the results from the sections 5.1 and 5.2 is discussed. In section 5.4, the Simon-Yule model is modified to fully capture the locality in the request sequence. In section 5.5, all the findings are summarized.

The simulation program follows the major steps described in Chapter 4. To start the process, some initial conditions are needed (Simon 1955). The initial condition we use is $f(1,3) = 3$, (i.e, three different records were used in the first three accesses). Throughout the analyses, the maximum number of accesses was 500.

5.1 Constant Entry Rates

With Simon's first assumption (i.e., constant entry rate ranging from 0.1 to 0.5), the average search cost - defined as the total number of probe records divided by the total number of accesses - for each heuristic is reported in Table 5.2 through Table 5.7. Figures 5.1 and 5.2 describe the results of the tables. For better understanding, however, several of tables and figures are summarized.

Many shows that request frequencies in many contexts obey Zipf's law. In addition, Knuth (1973) proved that the average search cost for Zipf's law is roughly D/lnD, where D is the number of distinct records. Table 5.1 shows how well the Simon-Yule model describes the usage pattern in terms of Zipf's law. For each α, the second column gives the cost of the optimal static ordering for requests from the Simon-Yule model. Comparing that column with the cost of Zipf's law (D/lnD) shows that the request frequencies from the Simon-Yule model is much closer to Zipf's distribution than to a uniform distribution (random ordering). It appears that the average search cost of optimal static ordering (A_0) is almost the same as Zipf's distribution when α is between 0.25 and 0.35. When α is less than 0.25, A_0 is less than that of Zipf's law. As α becomes progressively larger than 0.35, the gap between A_0 and Zipf's law also increases, but

Table 5.1: Comparisons Between Zipf's Law and the Simon-Yule Model (Constant α)

T	α = 0.1				α = 0.18				α = 0.25			
	D	Zipf's Law	OPT	RAND	D	Zipf's Law	OPT	RAND	D	Zipf's Law	OPT	RAND
100	19	6.45	4.03	9.5	27	8.19	6.08	13.5	30	8.82	6.57	15.0
200	24	7.55	4.81	12.0	42	11.23	8.48	21.0	57	14.09	11.23	28.5
300	31	9.03	5.64	15.5	56	13.91	10.16	28.0	83	18.78	16.38	41.5
400	41	11.04	6.38	20.5	75	17.37	12.69	37.5	106	22.73	21.34	53.0
500	55	13.72	7.74	27.5	99	21.54	16.09	49.5	139	28.17	27.12	69.5

T	α = 0.35				α = 0.4				α = 0.5			
	D	Zipf's Law	OPT	RAND	D	Zipf's Law	OPT	RAND	D	Zipf's Law	OPT	RAND
100	39	10.64	10.54	19.5	39	10.64	11.27	19.5	53	13.35	15.21	26.5
200	77	17.73	18.45	38.5	89	19.83	25.12	44.5	105	22.56	34.38	52.5
300	106	22.73	25.21	53.0	129	26.54	35.79	64.5	161	31.68	51.67	80.5
400	146	29.30	35.98	73.0	176	34.04	47.96	88.0	213	39.72	68.50	106.5
500	186	35.59	44.36	93.0	218	40.49	60.34	109.0	253	45.72	86.11	126.5

T: Total number of accesses
D: The number of distinct records
OPT: The average search cost of optimal static ordering under the Simon-Yule model.
RAND: The average search cost of random ordering

is still closer to Zipf's distribution than random ordering. This result clearly supports the performance evaluation of the heuristics with the Simon-Yule model.

From Table 5.2 to Table 5.7, we can observe several interesting facts.

(1) The heuristics are very effective. Under a randomly ordered list, the average search cost is D/2, where D is the number of distinct records in the list. The results show that, with the various values of alpha, all heuristics search less than half the list (which is the expected amount searched in a randomly ordered list).

(2) Figure 5.1 clearly shows that count always performs better than MTF and transpose, and its average cost is very close to that of optimal static ordering. On the average, the cost for count shows an increase of about 5% over optimal static ordering. For MTF, the increase is 20% - 31%; and for transpose, 10% - 22%. Count tends to converge to optimal static ordering as the number of accesses increases.

(3) Initially, MTF performs better, but as the number of accesses increases transpose works better (see Figure 5.2). This result is consistent with corollary 1 in Chapter 4 and clearly supports the need of a hybrid rule to optimize the heuristics.

Table 5.2 : The Average Search Costs for α = 0.1

t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
20	7	3.00	3.02	2.75	2.40	25.00	27.00	14.58
40	8	3.40	3.52	2.97	2.72	25.00	29.41	9.19
60	14	4.22	4.38	3.73	3.55	18.87	23.38	5.07
80	17	4.50	4.64	4.02	3.80	18.42	22.11	5.79
100	19	4.82	4.92	4.22	4.03	19.60	22.08	4.71
120	21	5.30	5.30	4.63	4.42	19.91	19.91	4.75
140	23	5.81	5.63	4.64	4.68	24.15	20.30	5.56
160	23	5.92	5.70	4.92	4.65	27.31	22.58	5.81
180	23	6.02	5.54	4.83	4.58	31.44	20.96	5.46
200	24	6.13	5.67	4.96	4.81	27.44	17.88	3.12
240	25	6.12	5.62	4.91	4.81	27.23	16.84	2.08
260	31	6.79	6.28	5.64	5.54	22.56	13.36	1.81
300	31	7.09	6.42	5.77	5.64	25.71	13.83	2.30
340	33	7.25	6.61	5.58	5.84	24.14	13.18	0.68
380	40	7.88	7.10	6.38	6.31	24.88	12.52	1.11
400	41	8.03	7.20	6.47	6.38	25.86	12.85	1.41
500	55	9.56	8.67	7.84	7.74	23.51	12.02	1.29

t = total number of accesses

D = number of distinct records

$Diff_1$ = the increase of MTF over optmal (%)

$Diff_2$ = the increase of transpose over optimal (%)

$Diff_3$ = the increase of count over optimal (%)

Table 5.3: The Average Search Costs for $\alpha = 0.18$

t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
20	8	3.20	3.30	3.12	3.00	6.67	10.00	4.13
40	13	4.13	3.92	3.80	3.75	10.43	4.53	1.33
60	20	5.67	5.63	5.32	5.20	9.04	8.27	2.31
80	26	6.79	6.61	6.39	6.26	8.47	5.59	2.08
100	27	6.69	6.49	6.16	6.08	10.03	6.74	1.32
120	31	7.68	7.27	6.89	6.75	13.78	7.70	2.07
140	35	8.44	8.01	7.52	7.31	15.46	9.58	2.87
160	38	9.34	8.75	8.06	7.72	20.98	13.34	4.40
180	38	9.43	9.22	7.96	7.62	23.75	21.00	4.46
200	42	10.39	10.07	8.81	8.48	22.52	18.75	3.89
240	48	11.29	10.91	9.49	9.14	23.52	19.37	3.83
280	52	11.89	11.36	9.86	9.69	22.70	17.23	1.75
300	56	12.47	11.92	10.38	10.16	22.74	17.32	2.17
340	63	13.55	12.72	11.13	10.82	25.23	17.56	2.87
380	67	14.23	13.52	11.68	11.52	23.52	17.36	1.39
400	75	15.42	14.73	12.88	12.69	21.51	16.08	1.50
500	99	19.26	18.40	16.25	16.09	19.70	14.36	0.99

t = total number of accesses

D = number of distinct records

$Diff_1$ = the increase of MTF over optmal (%)

$Diff_2$ = the increase of transpose over optimal (%)

$Diff_3$ = the increase of count over optimal (%)

Table 5.4: The Average Search Costs for $\alpha = 0.25$

t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	0.00	0.00	0.00	0.00
20	8	3.20	3.30	3.10	3.00	6.67	10.00	3.33
40	13	4.13	3.92	3.80	3.45	19.71	17.97	10.14
60	21	6.15	6.27	5.90	5.22	17.82	20.11	13.03
80	26	7.10	7.15	6.61	6.12	16.01	16.83	8.01
100	30	7.73	7.58	7.03	6.57	17.66	15.37	7.00
120	35	8.62	8.57	7.92	6.98	23.50	22.78	13.46
140	45	10.89	10.64	10.04	9.01	20.87	18.09	11.43
160	49	11.71	11.44	10.81	9.78	19.73	16.97	10.53
180	56	13.68	13.31	12.54	11.34	20.63	17.37	10.58
200	57	14.11	13.67	12.51	11.23	25.65	21.73	11.40
240	69	16.81	16.19	14.89	13.23	27.06	22.37	12.55
280	78	18.51	18.01	16.45	15.01	23.32	19.99	9.59
300	83	19.51	18.82	17.22	16.38	19.11	14.90	5.13
340	92	21.82	20.94	18.94	18.04	20.95	16.08	4.99
380	102	24.40	23.55	21.00	19.99	22.06	17.81	5.05
400	106	25.33	24.44	21.66	21.34	18.70	14.53	1.50
500	139	32.10	31.03	27.75	27.12	18.36	14.42	2.32

Table 5.5: The Average Search Costs for $\alpha = 0.35$

t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
20	12	5.56	5.58	5.40	5.20	7.50	7.88	3.85
60	17	6.22	6.85	5.80	5.41	14.97	26.62	7.21
80	36	11.97	11.95	11.09	10.11	18.40	18.20	9.69
100	39	12.58	12.06	11.15	10.56	19.13	14.20	5.69
120	47	14.34	14.32	13.10	12.34	16.21	16.05	6.16
140	57	16.85	16.74	15.55	14.65	15.02	14.27	6.14
160	65	19.01	18.81	17.60	16.81	13.09	11.90	4.70
180	71	20.81	20.49	19.26	18.21	14.28	12.52	5.77
200	77	22.85	22.20	20.96	19.84	15.17	11.90	5.65
240	88	25.87	25.12	23.41	22.36	15.70	12.34	4.70
280	101	29.20	28.90	26.40	24.91	17.22	16.02	5.98
300	106	30.44	30.18	27.34	25.86	17.71	16.71	5.72
340	121	34.59	34.34	31.06	29.81	16.03	15.20	4.19
380	137	38.84	38.59	34.91	33.74	15.12	14.37	3.47
400	146	41.23	40.69	37.12	35.98	14.59	13.09	3.17
500	186	52.55	52.48	46.90	46.02	14.19	14.04	1.91

Table 5.6: The Average Search Costs for $\alpha = 0.40$

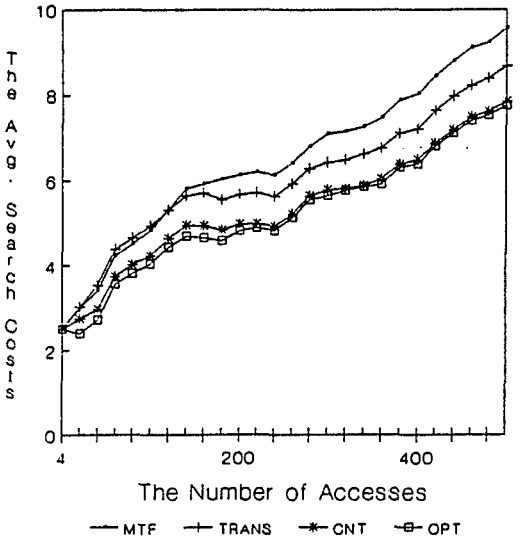
t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
20	12	5.75	6.00	5.25	5.00	15.00	20.00	5.00
40	17	6.32	6.20	5.67	5.34	18.35	16.10	6.18
60	29	9.53	9.57	9.03	8.12	17.36	17.86	11.21
80	36	12.00	12.04	11.20	10.02	19.76	20.16	11.78
100	39	14.19	13.79	12.49	11.27	25.91	22.36	10.83
120	52	17.57	17.18	15.88	13.97	25.77	22.98	13.67
140	63	20.61	20.74	19.02	17.75	16.11	16.85	7.15
160	73	23.82	23.52	21.82	20.34	17.11	15.63	7.28
180	81	26.22	26.44	24.11	22.85	14.75	15.71	5.51
200	89	28.75	28.95	26.45	25.12	14.45	15.25	5.29
240	102	32.48	32.70	29.45	28.13	15.46	16.25	4.69
280	123	38.96	40.16	36.11	34.28	13.65	17.15	5.34
300	129	40.01	42.37	37.54	35.79	11.79	18.39	4.89
340	148	47.16	48.46	43.27	41.11	14.72	17.88	5.25
380	164	52.69	54.28	47.01	45.13	16.75	20.27	4.17
400	176	56.69	57.90	50.17	47.96	18.20	20.73	4.61
500	218	71.12	72.55	62.25	60.34	17.87	20.24	3.17

Table 5.7: The Average Search Costs for $\alpha = 0.50$

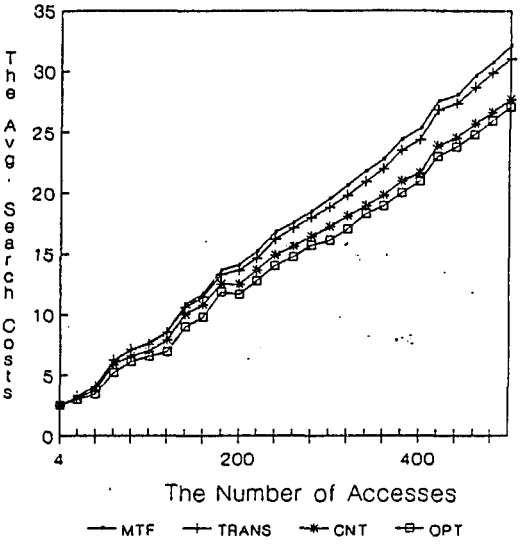
t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
20	15	7.00	7.10	6.75	6.11	14.57	16.20	10.47
40	19	7.77	8.50	7.13	6.67	16.49	27.44	6.90
60	32	11.37	11.82	10.01	9.45	20.32	25.08	5.93
80	45	15.61	16.15	15.16	13.31	17.28	21.34	13.90
100	53	18.36	18.73	17.56	15.21	20.71	23.14	15.45
120	64	21.72	22.07	20.72	18.45	17.72	19.62	12.30
140	75	25.39	26.11	24.39	22.01	15.36	18.63	10.81
160	87	29.74	30.40	28.59	26.28	13.17	15.68	8.79
180	97	32.91	33.31	31.52	28.83	14.15	15.54	9.33
200	105	36.20	36.97	34.38	32.12	12.70	15.10	7.04
240	128	43.02	43.71	40.55	37.78	13.87	15.70	7.33
280	149	51.35	51.41	47.26	44.55	15.26	15.40	6.08
300	161	56.41	56.57	51.67	48.74	15.74	16.06	6.01
340	182	64.76	65.91	59.22	57.15	13.32	15.33	3.62
380	203	71.40	72.79	66.06	64.63	10.48	12.63	2.21
400	213	75.37	76.37	68.50	66.24	13.78	15.29	3.41
500	253	88.09	90.01	83.09	84.14	8.69	10.80	2.34

Figure 5.1: The Average Search Costs (Constant Entry Rates)

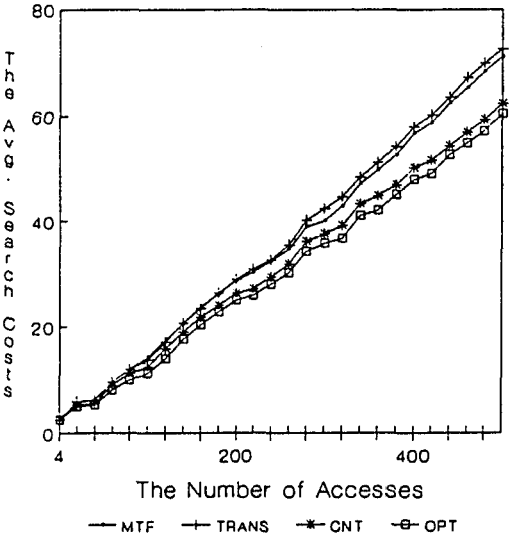
Alpha = 0.1



Alpha = 0.25



Alpha = 0.40



Alpha = 0.5

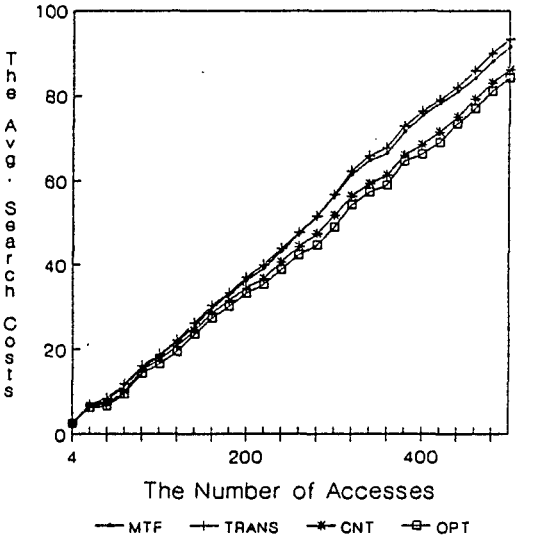
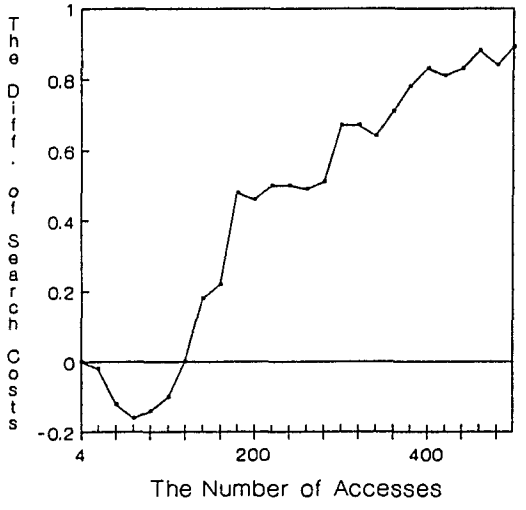
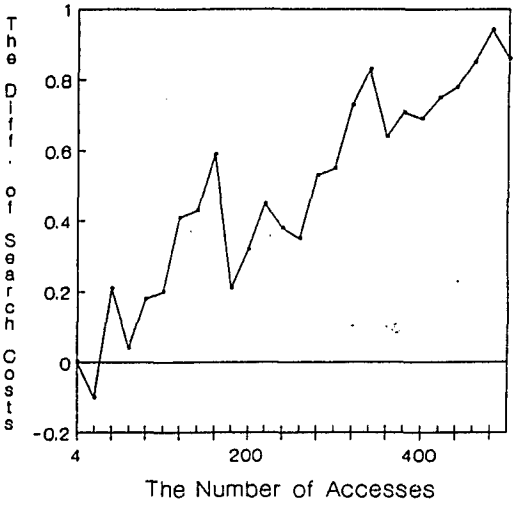


Figure 5.2: Effects of Alpha (Constant Entry Rates)

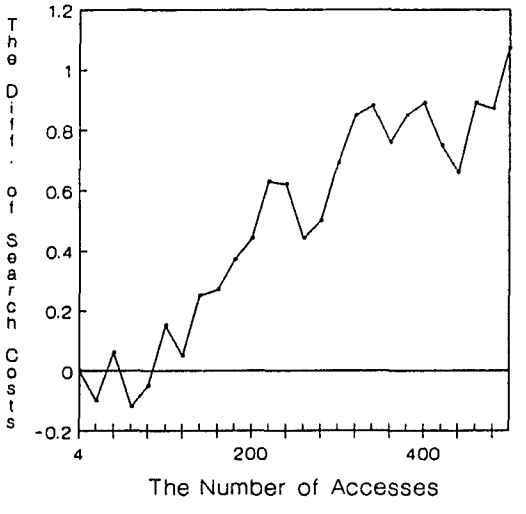
Alpha = 0.1



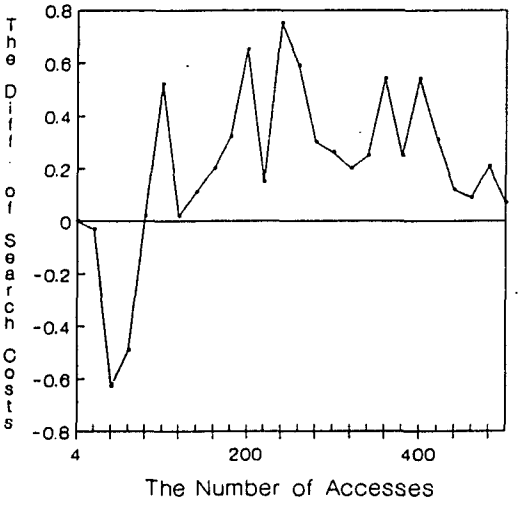
Alpha = 0.18



Alpha = 0.25



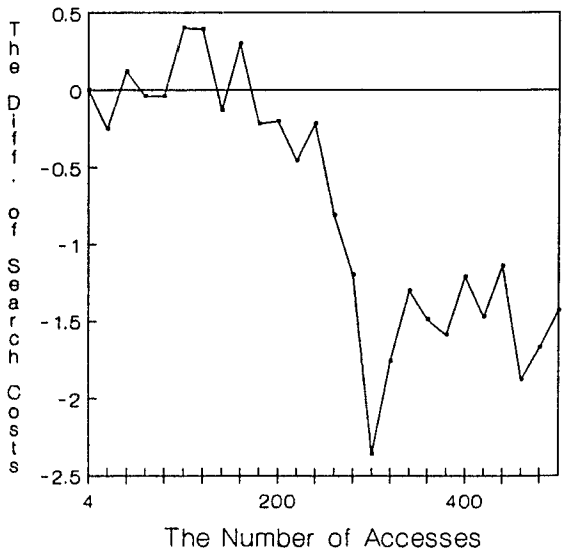
Alpha = 0.35



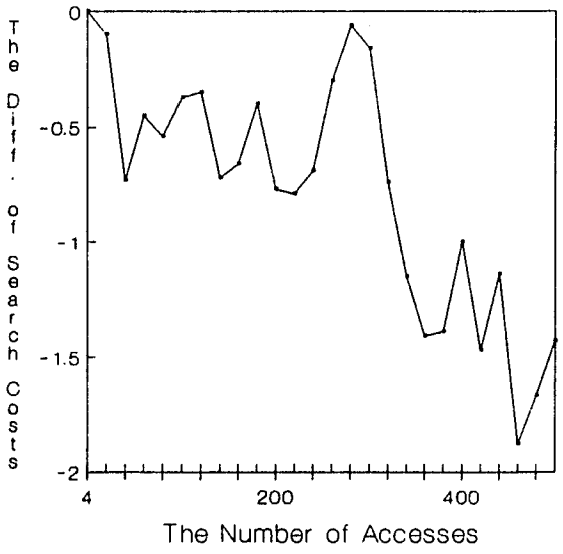
* The difference of search costs = MTF - Trans

Figure 5.2.1: Effects of Alpha (Constant Entry Rates)

Alpha = 0.4



Alpha = 0.5



*** The difference of search costs = MTF - Trans**

In summary, the simulation results seem to be consistent with the asymptotic approach rather than the amortized analysis, except that the performance between MTF and transpose depends on the value of α: (1) $A_M \leq 2A_O$; (2) $A_C = A_O$; and (3) $A_M \geq A_T$ if $\alpha \geq 0.4$ and $A_M \leq A_T$ if $\alpha \leq 0.4$. Clearly, count is the best of the heuristics considered here. However, since count requires substantial additional space for counter fields, it cannot be considered a memoryless heuristic like MTF and transpose. Unless the counter fields can be used for other purposes, either MTF or transpose should be considered as an optimal heuristic.

The question then arises as to which heuristic is better, MTF or transpose? Figures 5.2 and 5.2.1 clearly show that neither MTF nor transpose outperforms all the time over each other. We conclude that it is better to use MTF and then to switch to transpose as the number of accesses increases. If α is larger than 0.4, however, MTF tends to work better than transpose all the time.

Effects of A Moving Distance

MTF and transpose is the relatively extreme case of a moving distance for an accessed record. If k is the distance to the front, MTF is move-ahead-k and transpose is move-ahead-1. Thus, the effects of an intermediate moving distance for an accessed record as parameters vary would be

interesting. The simulation was conducted with the various constant entry rates and moving distances ranging from 2 to 7. As usual, if the distance to be moved exceeds the distance to the front, the record is only moved to the front.

The result shown in Table 5.8 indicates that there is no optimal value of k, but the best value of k can be determined by the values of alpha. The overall trend also shows that initially (i.e., $\alpha \geq 0.5$) move an accessed record with a large distance and as the number of accesses increases (i.e., $\alpha = 0.1$), reduce the moving distance.

Table 5.9: The Effects of a Moving Distance

The Value of α	The Best k
0.1	2
0.18	3
0.25	5
0.35	6
0.4	7
0.5	7
0.7	7
0.9	7

5.2 Decreasing Entry Rates

It is well known that the entry rate α decreases as the number of accesses increases (Simon 1955). For a more realistic performance analysis, therefore, the simulation should be experimented with the decreasing entry rate α. In Chapter 2, we stated that Simon refined his basic model (i.e., constant entry rates) and proposed the new model assuming decreasing entry rate α. The simulation was conducted with three different cases of the decreasing entry rate. The results using Simon's refined model (decreasing entry rates) are shown in Tables 5.9 through 5.11. Again, Table 5.1.1 shows that the Simon-Yule model also describes the usage pattern very well with the decreasing entry rates. That is, the average search cost of optimal static ordering is almost the same as that of Zipf's distribution.

Figures 5.3 and 5.4 indicate that the results are consistent with the constant entry case; heuristics are very effective; count is superior to MTF and transpose; and MTF performs better initially, but as the number of accesses increases (i.e., α decreases), transpose outperforms MTF.

5.3 A Proposed Hybrid Rule

As discussed earlier, Bitner (1979) suggests a hybrid rule and a switching time from MTF to transpose based solely on the number of accesses, which is unreasonable. For practical purposes, therefore, we need clearer and more

Table 5.1.1: Comparisons Between Zipf's Law and the Simon-Yule Model (Decreasing α)

T	Case 1				Case 2				Case 3			
	D	Zipf's Law	OPT	RAND	D	Zipf's Law	OPT	RAND	D	Zipf's Law	OPT	RAND
100	30	8.82	6.87	15.0	38	10.45	9.81	19.0	39	10.64	11.98	19.5
200	43	11.43	8.74	21.5	52	13.16	11.95	26.0	45	11.82	12.21	22.5
300	56	13.91	9.95	28.0	63	15.21	12.91	31.5	52	13.16	13.01	26.0
400	73	17.01	11.89	36.5	78	17.90	14.98	39.0	61	14.84	13.78	30.5
500	96	21.03	15.01	48.0	100	21.71	18.57	50.0	74	17.19	15.11	37.0

Case 1: $\alpha = 0.5t^{-0.2}$

Case 2: $\alpha = 0.386$, if $t \leq 50$
$\alpha = 0.217$, if $50 \leq t < 100$
$\alpha = 0.160$, if $100 \leq t < 150$
$\alpha = 0.204$, if $150 \leq t < 200$
$\alpha = 0.160$, if $200 \leq t < 250$
$\alpha = 0.139$, if $250 \leq t$

Case 3: $\alpha = 0.386$, if $t \leq 100$
$\alpha = 0.1$, otherwise

T: Total number of accesses
D: The number of distinct records
OPT: The average search cost of optimal static ordering under the Simon-Yule model
RAND: The average search cost of random ordering

Table 5.9: The Average Search Costs for Decreasing α

(Case 1)

t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
20	8	3.20	3.35	3.05	3.00	6.67	11.67	1.67
40	13	4.10	4.20	3.77	3.75	9.33	12.00	0.53
60	21	6.13	6.28	5.88	5.67	8.11	10.76	3.70
80	26	7.07	7.09	6.55	6.45	9.61	9.92	1.55
100	30	7.62	7.48	6.97	6.87	10.92	8.88	1.46
120	32	8.08	7.92	7.22	7.08	14.12	11.86	1.98
140	38	9.25	9.08	8.31	8.15	13.50	11.41	1.96
160	41	10.21	9.90	8.96	8.75	16.69	13.14	2.40
180	41	10.42	10.20	8.76	8.51	22.44	19.86	2.94
200	43	10.77	10.35	9.02	8.74	23.23	18.42	3.20
240	48	11.30	11.39	9.55	9.28	21.77	22.74	2.91
280	52	11.89	11.94	9.95	9.69	22.70	23.22	2.68
300	56	12.46	12.52	10.48	9.95	25.23	25.83	5.33
340	60	13.04	12.96	10.76	10.48	24.43	23.66	2.67
380	65	13.87	13.66	11.38	10.91	27.13	25.21	4.31
400	73	15.08	14.74	12.53	11.89	26.83	23.97	5.38
500	96	18.70	18.15	15.78	15.01	24.58	20.92	5.13

* Decreasing rate of α

$$\alpha = 0.50 * t^{-0.2}$$

Table 5.10: The Average Search Costs for Decreasing α

(Case 2)

t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
50	8	7.80	7.82	7.24	6.98	11.75	12.03	3.72
100	13	11.37	11.48	10.47	9.81	15.90	17.02	6.73
200	21	14.93	14.83	12.75	11.98	24.62	23.79	6.43
300	63	16.31	16.29	13.72	13.01	25.37	25.21	5.46
400	78	19.08	18.67	15.65	14.98	27.37	24.63	4.47
500	102	23.27	22.28	19.21	18.57	25.31	19.98	3.45

* Decreasing rate of α

$\alpha = 0.386$, if $t < 50$
$\alpha = 0.217$, if $50 \leq t < 100$
$\alpha = 0.160$, if $100 \leq t < 150$
$\alpha = 0.204$, if $150 \leq t < 200$
$\alpha = 0.160$, if $200 \leq t < 250$
$\alpha = 0.139$, if $250 \leq t$

Table 5.11: The Average Search Costs for Decreasing α

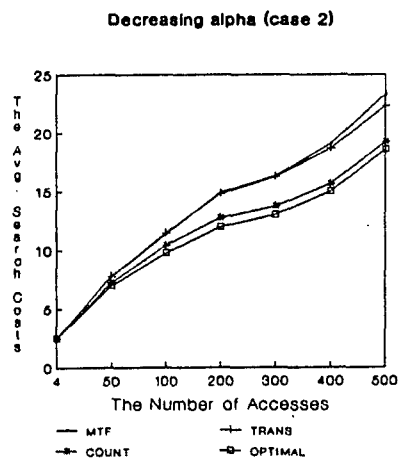
(Case 3)

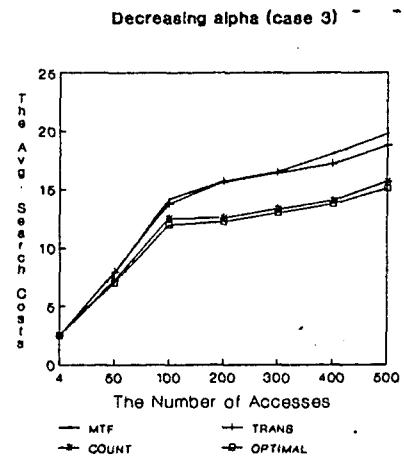
t	D	MTF	Trans	Count	Optima	$Diff_1$	$Diff_2$	$Diff_3$
4	4	2.50	2.50	2.50	2.50	0.00	0.00	0.00
50	23	7.82	7.94	7.24	6.98	12.03	13.75	3.72
100	39	14.19	13.79	12.49	11.98	18.45	15.11	4.26
200	45	15.69	15.65	12.59	12.21	28.50	28.17	3.11
300	52	16.53	16.45	13.32	13.01	27.06	26.44	2.38
400	61	18.05	17.21	14.08	13.78	30.99	24.89	2.18
500	74	19.77	18.79	15.71	15.11	30.84	24.35	3.97

* Decreasing rate of α

$\alpha = 0.386$, if $t \leq 100$
$\alpha = 0.1$, if $t > 100$

Figure 5.3: The Average Search Costs (Decreasing Alphas)





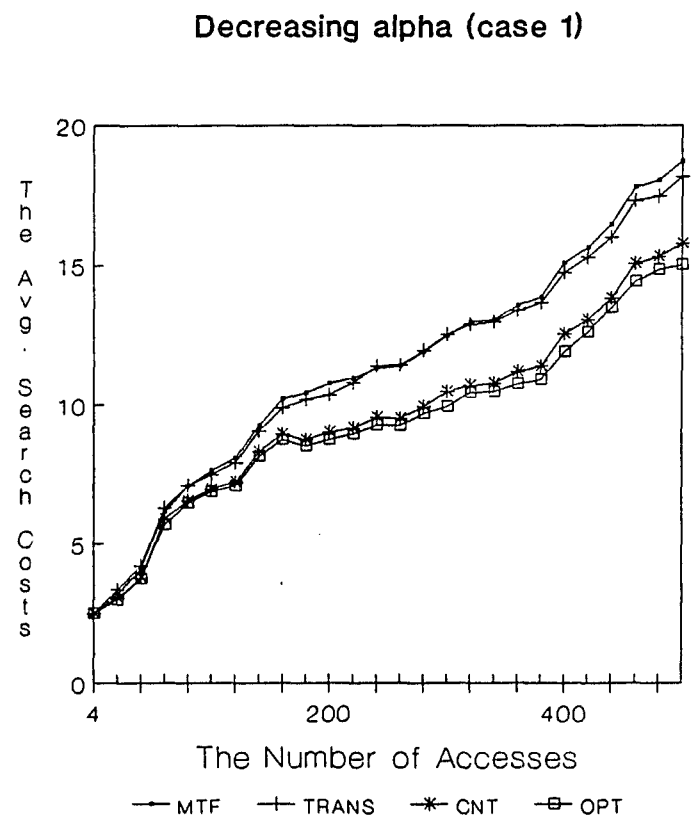
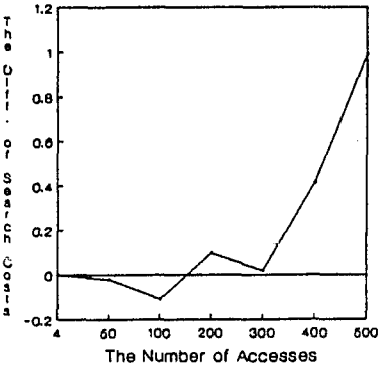
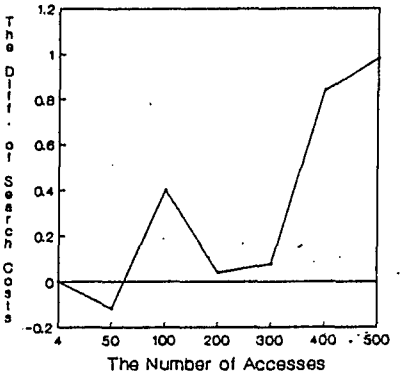


Figure 5.4: Effects of Alpha (Decreasing Entry Rates)

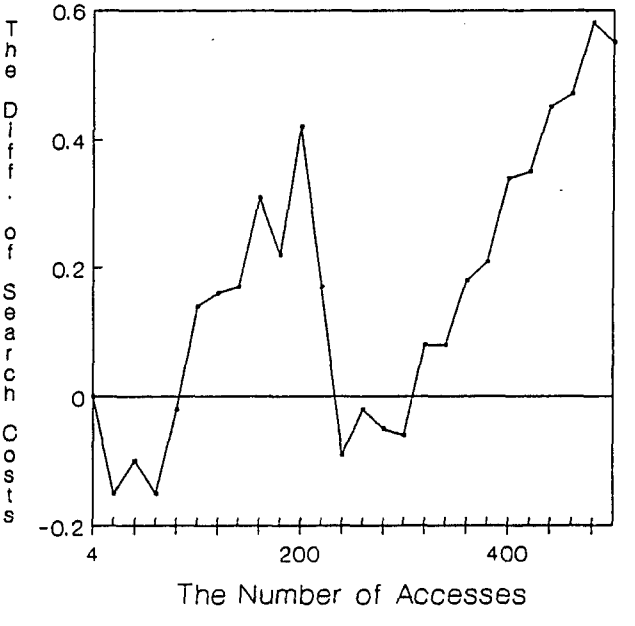
(case 2)



(case 3)



(case 1)



• The Difference of Search Costs = MTF - Transpose

reasonable guidelines concerning switching time (Hester and Hirschberg 1986).

From Figures 5.2, 5.2.1, and 5.4, we derive a more reasonable switching time. Figures 5.2 and 5.2.1 show that when α is larger than 0.35, MTF tends to outperform transpose. However, when α is less than or equal to 0.35, the performance depends on the number of accesses; if the number of accesses is approximately larger than 110, transpose performs better than MTF. With the cases of decreasing α shown in Figure 5.4, we also observe the same phenomenon. Since the difference between MTF and transpose tends to be narrow down again for $\alpha = 0.35$, however, the simulation study with $t = 1000$ is conducted. The results shown in Table 5.12 and Figure 5.2.2 indicates that for $\alpha = 0.35$ there is some fluctuation on the performance between MTF and transpose, but for $\alpha \geq 0.4$ MTF tends to outperform transpose after the number of accesses exceeds 180. Therefore, the proposed switching time is to switch from MTF to transpose if $\alpha \leq 0.40$ and $t \geq 180$.

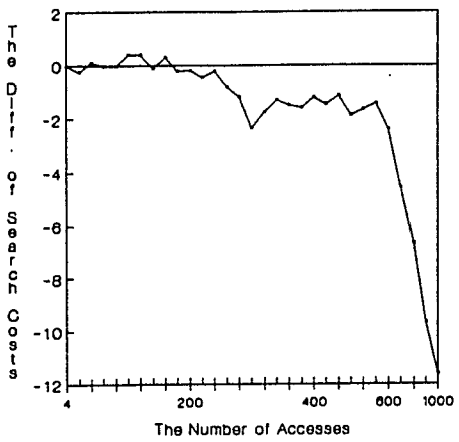
This switching time is more reasonable because it additionally incorporates the steady-state condition of the list, which can be determined by a value of alpha. Since the entry rate α can be easily obtained by dividing the number of distinct records in the list by the number of accesses, no significant overhead of a hybrid rule would be

Table 5.12: The Average Search Costs of MTF and Transpose Until t = 1000

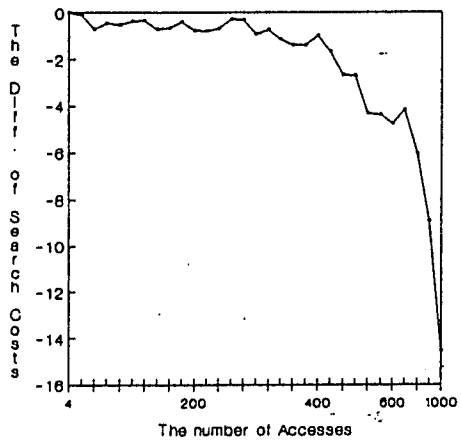
t	α = 0.35		α = 0.4		α = 0.5	
	MTF	Trans	MTF	Trans	MTF	Trans
600	62.62	62.01	84.42	86.82	109.01	113.77
700	72.37	74.03	96.61	101.17	129.08	133.25
800	80.91	83.94	109.03	115.71	148.41	154.45
900	91.25	94.08	118.43	128.13	165.17	174.06
1000	99.70	104.68	131.43	143.04	178.45	192.99

Figure 5.2.2: Effects of Alpha (t = 1000)

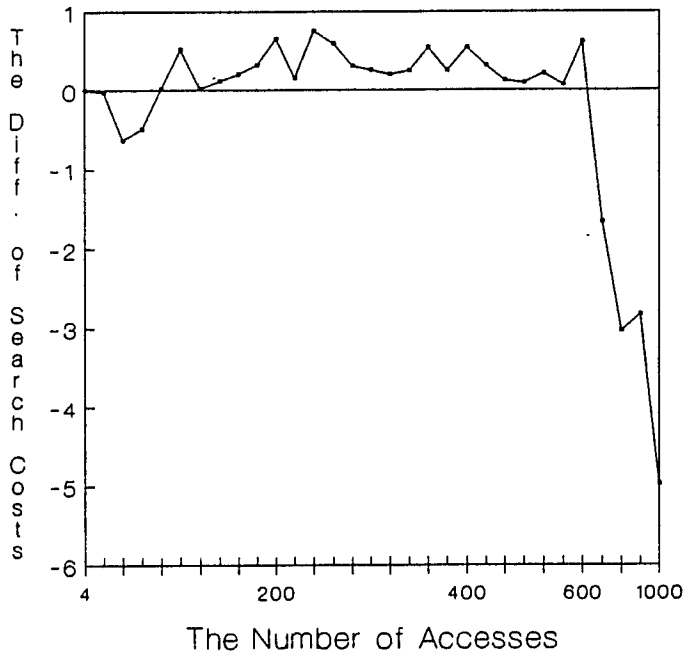
Alpha = 0.4



Alpha = 0.5



Alpha = 0.35



* The difference of search costs = MTF - Trans

incurred. Using this scheme, we may obtain a much better performance than with the application of either MTF or transpose only. If the number of accesses is relatively small compared with the number of records (i.e., a large α or $t < 180$), however, applying MTF only is preferred. If the number of accesses is large enough to offset the advantage of a hybrid rule, using transpose only is attractive.

5.4 Capturing Locality

As discussed in Chapter 3, if there is a strong locality in the request sequence, MTF performs better than transpose all the time (Bently and McGeoch 1985, Sleater and Tarjan 1985, Bellow 1987). Bently and McGeoch (1985) compared the performance of MTF and transpose by using words in four pascal files and six English text files as input data, indicating that MTF always outperforms transpose.

The proposed Simon-Yule approach seems that it does not fully capture the locality phenomenon in the request sequence, although it relaxes the previous unrealistic assumptions (i.e., independent accesses and fixed access probability). In order to model the locality, therefore, the proposed approach is modified (specifically STEP 3) and applied to the performance analyses for MTF and transpose.

In STEP 3, a specific record is chosen from the records of the group determined in STEP 2 with an equal probability. Then the record is moved to the very last position of the list of the next higher group. Suppose that a list is obtained as follows and a record 5 is chosen as an accessed record in STEP 3:

1 2 3/ 4 **5** 6/ 7 8 9/ 10 11 12 13 (5.1)

Number of Occurrences = (4) / (3) / (2) / (1)

Since the record 5 is accessed four times, it is necessary to keep this information to generate a next accessed record and move it to the very last position of the next group. The result of updating the list is as follows:

1 2 3 **5**/ 4 6/ 7 8 9/ 10 11 12 13 (5.2)

Then the list (5.1) is arranged according to the corresponding heuristic in STEP 4. For example, if MTF is applied, the list would be:

5 1 2 3 4 6 7 8 9 10 11 12 13 (5.3)

In summary, the very recently accessed record for each group is the very last one. Since locality shows that there is a strong correlation between most recently accessed records and a future access, it is logical to choose the very last record as an accessed one in STEP 3 in order to

capture the locality phenomenon. The STEP 3, thus, is revised as follows.

[STEP 3']: Find a specific record from the group of records chosen in STEP 2

Choose the very last one within the group chosen in STEP 2.

The empirical study is also performed to support how well the modified approach captures the locality phenomenon. Three pascal and two English text files are used as input data. The results summarized in Table 5.13 show that MTF performs better than transpose and the simulation results using the modified approach is also consistent with the empirical study.

It seems that the modified approach reasonably captures the locality phenomenon in the request sequence. It is also confirmed that MTF performs better than transpose if the search sequence involves a strong locality.

5.5 Summary

Using the Simon-Yule model, we conducted a more reasonable and realistic performance evaluation of self-organizing linear search heuristics. The analyses focus mainly on the representative of a large section: MTF,

transpose, count, and move-ahead-k. The simulation was performed with the constant entry rates ranging from 0.1 to 0.5 and with the decreasing entry rates. The results from both constant and decreasing entry rate are the same and are consistent with the asymptotic approach; i.e.,

- The heuristics are effective.
- Count is superior to MTF and transpose, which is close to optimal static ordering. Therefore, if counter fields would be needed for a special purpose or storage space is not limited, count would be the best.
- The average search cost of MTF is less than twice that of optimal static ordering. Specifically, its theoretical bound under the Simon-Yule model is 58% increase over that of optimal static ordering.

We also find some exceptions. Initially, MTF performs better, but later transpose outperforms MTF. Using the number of accesses and the entry rate, we suggest the more reasonable switching time from MTF to transpose without the significant overhead of a hybrid rule. The specific guideline suggests a switch from MTF to transpose if ($\alpha \leq 0.40$) and ($t \geq 180$).

The Simon-Yule approach is also modified in order to fully capture a strong locality phenomenon in the request sequence. The modified approach shows that it captures the locality phenomenon very well. It also provides the

possible way for estimating the expected search cost of the heuristics under the assumption of the strong locality in the request sequence.

In conclusion, we find that the results seem to be consistent with the asymptotic approach except for the performance between MTF and transpose. Since MTF performs better initially and later transpose outperforms MTF, we suggest a more reasonable switching time, which would be valuable information for practitioners. However, it is also confirmed that MTF performs always better than transpose if there is a strong locality in the request sequence.

Table 5.13: Simulation Results Using The Modified Approach

Input	t	d	Actual Data		Simulation Results	
			MTF	Trans	MTF	Trans
P1	405	83	23.89	34.78	18.34	23.76
P2	563	133	36.07	42.46	30.85	42.17
P3	703	158	42.63	58.26	35.16	51.15
T1	1048	500	147.60	162.68	172.23	225.76
T2	847	389	95.23	126.41	115.34	162.78

* P1, P2, P3 are PASCAL files

* T1 : "The Injustice of the Death Penalty," written by Neal Devins and R. B. Herron, The Christian Science Monitor, 1983.

* T2 : "A Draft Isn't Needed," written by D. Bandow, The New York Times, 1982.

CHAPTER 6

CONTINUOUS SPEECH RECOGNITION: AN APPLICATION

Recent study of continuous speech recognition (CSR) in the artificial intelligence arena has called for the use of statistical models of text (Jelinek et al. 1983, Young et al. 1989, White 1990). A major issue in this field is the lack of effective and objective evaluation of the models as well as a more adaptive framework for CSR. In Chapter 2, we evaluate the statistical models of text and identify the Simon-Yule model is the most promising one. From Chapter 3 to Chapter 5, it is also found that count is superior to MTF and transpose. If counter fields are needed for some special purposes or storage space is not limited, count would be the best choice. In this chapter, thus, we will show how the two findings can be incorporated into a UDIS design like CSR. Based on Simon's explanatory processes of imitation and association, we suggest an adaptive framework for CSR. Furthermore, a self-organizing mechanism incorporating count rule is developed for a statistical language model for CSR.

6.1 Continuous Speech Recognition (CSR)

The basic CSR system where sentences are produced continuously in a natural manner consists of an acoustic processor (AP) followed by a linguistic decoder (LD) as shown in Figure 6.1. Traditionally, the AP is designed to act as a phonetician, transcribing the speech waveform into a string of phonetic symbols, while the LD translates the possible garbled phonetic string into a string of words.

In Figure 6.2 speech recognition is formulated as a problem in IBM's communication theory view (Jelinek et al. 1983). The IBM's approach combines the speaker and acoustic processor into an acoustic channel. The speaker is transforming the text into a speech waveform and the acoustic processor is acting as a data transducer and compressor.

6.2 Statistical Models of Text

The AP produces an output string y. From this string y, the linguistic decoder (LD) makes an estimate $w\hat{}$ of the word string w produced by the text generator (see. Figure 6.2). To minimize the probability error, $w\hat{}$ must be chosen so that $P(w\hat{}|y) = max\ P(w|y)$.

By Bayes' rule:

$$P(w|y) = \frac{P(w)P(y|w)}{P(y)} \qquad (6.1)$$

Let $P(w,y)$ be the probability of the joint observation of the input-output pair w and y. Since $P(y)$ does not depend on w and thereby maximizing $P(w|y)$ is equivalent to maximizing the likelihood $P(w,y) = P(w)P(y|w)$, the goal of the linguistic decoder is to find that word string $w^$ which maximizes $P(w,y)$. Let $P(w)$ be the probability that w was generated by the text generator and $P(y|w)$ the probability that the acoustic processor output the word string y after the speaker read w. To estimate $P(w)$ and $P(y|w)$, the LD requires two models (Jelinek et al. 1983): (1) a statistical language model of text which provides the information about which words are most probable with respect to previous other words in the word string w, and (2) an acoustic channel model which provides the information about which words are most probable based on a sound string w read by the speaker. Once the information for computing $P(w)$ and $P(y|w)$ is available from the two models, it is possible for the LD to compute the likelihood of each sentence in language and determine the most likely $w^$ directly. In this study, we focus on the statistical language model only.

Figure 6.1: A Continuous Speech Recognition System.

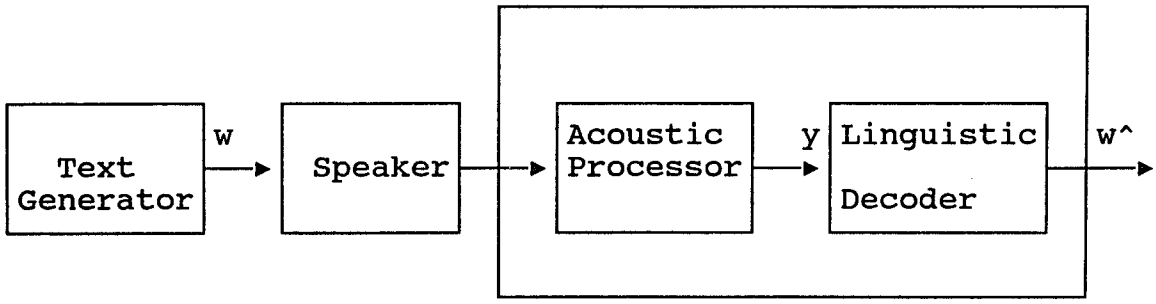
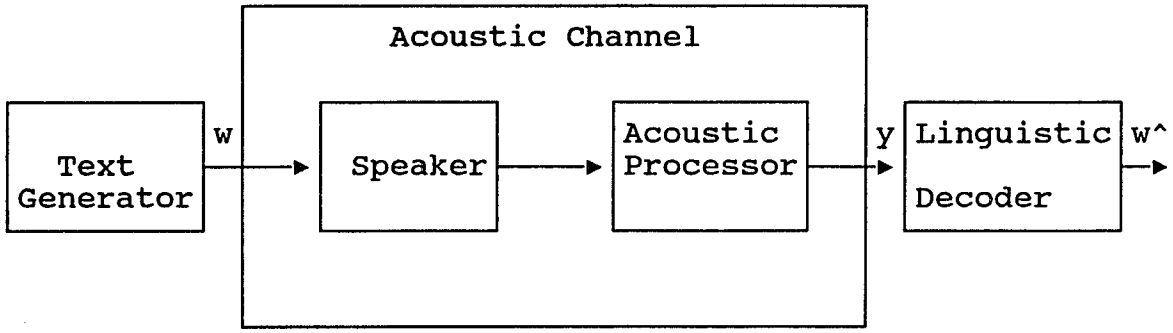


Figure 6.2: The IBM Approach of Continuous Speech Recognition (Jelinek et al. 1983).



6.3 Evaluating the Statistical Models of Text

In the IBM approach, the k^{th} order (or k-gram) Markov chain is used for a statistical language model for CSR. Particulary, k = 2 (i.e., trigram Markov chain model) is used. However, there has been some negative aspects about trigram Markov chain model. Jelinek (1985) pointed out that there is "nothing to recommend the trigram language model (k = 2) except its simplicity and ease of construction from training text." He also pointed out that "the selection of the exact classification scheme (for the conditional words), and its use in determining a large amount of text, is an unsolved problem that will claim increasing attention of researchers." These comments indicate the need for an effective and objective evaluation procedure for statistical models of text.

As discussed in Chapter 2, there are several other statistical models of text which were originally proposed for explaining the empirical phenomenon identified by Zipf (1949), i.e., if one takes the words making up an extended body of text and ranks them by their number of occurrences, then for each word the rank r multiplied by its corresponding frequency of occurrence will be approximately constant. A more simplified version of the Markov chain model is the multinomial urn model which assumes that (1)

the author's vocabulary is fixed, and (2) the probability of using each word is fixed.

A further generalization of the Markov chain model was proposed by Simon in 1955. Simon viewed the choice of modes of language as a two-fold process: by imitation and association. According to Simon, an author writes by process of imitation: sampling segments of word sequences from other words he has written, from words of other authors, and from segments he has heard. Suppose that there are two texts, A and B. In text A, the word "He", which occurs 1000 times and ranks 20th, has very nearly the same rank - 21th - in text B. A more interesting observation is that, of the 100 most frequent words in text A, 78 are among the top 100 in text B. Simon explained that the similarity in ranking of "common" words is due to the process of imitation.

An author writes also by process of association: sampling earlier segments of the word sequences. In text A, the proper noun "Bloom" occurs 926 times and ranks 30th in frequency. Simon argued that if the author had named the proper noun as "Smith", that noun, instead of "Bloom", would have ranked 30th.

In summary, Simon (1955) believed that both imitation and association will tend to dictate the occurrence of a

particular word with a probability somewhat proportional to its frequency of occurrence in the language, and to its previous frequency of use by a communicator.

6.4 Implication of Imitation for CSR

As discussed earlier, the continuous speech recognition system illustrated in Figure 6.2 indicates that the LD requires a statistical language model to provide the probability, *P(w)*, that the text generator produces a word string w. If the text used is artificial, e.g., generated by the grammar of a Raleigh language (Jelinek et al. 1983), then the computation of *P(w)* is relatively easy. For natural texts, however, the computation is much more difficult.

The trigram Markov chain model might be the most common approach for providing *P(w)* for natural text (Jelinek 1985). In such a model,

$$P(w) = \prod_{i=1}^{n} P\,(w_i \mid w_{i-1}, w_{i-2}), \qquad (6.2)$$

where $w = w_1, w_2, \ldots, w_n$, denoting a string of n words. By analyzing a large set of training sentences a matrix of word transition probabilities $P(w_i \mid w_{i-1}, w_{i-2})$ is constructed (Young et al. 1989). Specifically, the transition probabilities are constructed by counting the number of

times the words w_{i-2}, w_{i-1} occur adjacent to each other, without regard to all the histories resulting in the same two words.

Depending on the training sentences used, the estimated probability *P(w)* represents the frequency of occurrence of w in the language. Thus, if the training sentences are from an office text, then *P(w)* is the chance that the sentence w is written or spoken in the office environment. The process of estimating equation (6.2) is based on the assumption that the choice of any sentence depends, in large measure, upon the choices of other writers or speakers. This assumption is consistent with Simon's process of imitation involving language communication.

6.5 Implication of Association for CSR

According to Simon, language production also involves the process of association, because a writer has a tendency to sample earlier segments of his/her writing. As we can see from Table 6.1, the productivity of an old word at a certain point of time is roughly proportional to the number of its previous appearances, which is consistent with the concept of the rich-get-richer. The process of association implies that the probability *P(w)* in equation (6.2) will be dynamic, adaptive, and increasing as the same sentence is

repeated in the course of communication. Thus, we suggest in Figure 6.3 a revised version of Figure 6.2. This new framework of a continuous speech recognition system adapts to a speaker's usage pattern by attempting to constantly update the probability *P(w)* in the text generator, thus generating texts in a more natural manner.

Figure 6.3: A Revised Version of Figure 6.2 Based on Simon's Process of Association.

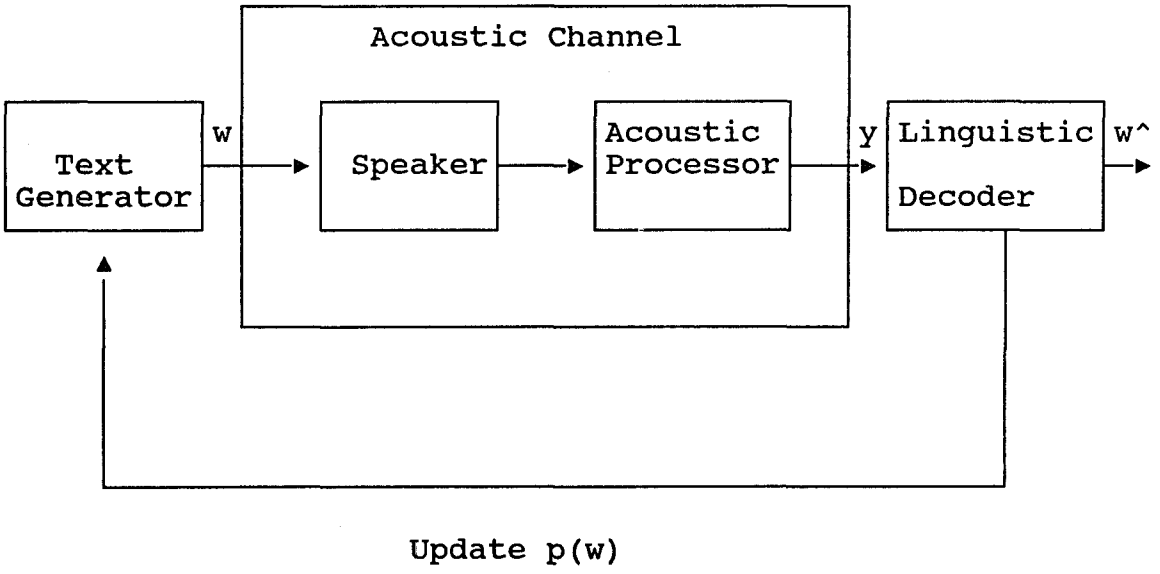


Table 6.1: The Realization of the Sentences.

Example of Text:

"Text modeling is the study of how a literary text is generated. The field is a relatively old discipline that has taken on new life due to recent developments in artificial intelligence."

Time Series Representation:

		Sum
Text	1 1	2
modeling	1	1
is	1 1 1	3
the	1 1	2
study	1	1
of	1	1
how	1	1
a	1 1	2
literary	1	1
generated	1	1
field	1	1
relatively	1	1
old	1	1
discipline	1	1
that	1	1
has	1	1
taken	1	1
on	1	1
new	1	1
life	1	1
due	1	1
to	1	1
recent	1	1
developments	1	1
in	1	1
artificial	1	1
intelligence	1	1
Sum = 27		32

No. of Types = 27 No. of Tokens = 32

6.6 A Self-Organizing Language Model

In this section, we develop the self-organizing mechanism for the proposed adaptive framework for CSR by incorporating count rule. Since text files tend to represent a strong locality and MTF performs better on the locality phenomenon, MTF might be a right choice for this case. However, a most important factor for a statistical language model in the proposed adaptive framework is to provide a detailed mechanism for updating dynamic and adaptive probabilities $P(w)$ constantly. Clearly, MTF would not the right choice for this purpose, because it does not keep any information of the usage frequencies due to its memoryless characteristics.

As discussed in Chapter 5, count performs bettern than MTF and transpose. Its only disadvantage is to take some storage space for counter fields. If counter fields are needed for a special purpose or storage space is not an important design issue, however, count would be the best. Since we need some mechanisms to keep track of usage frequencies for each word in order to update $P(w)$, which are provided by count, count is recommended to be incorporated into a statistical language model.

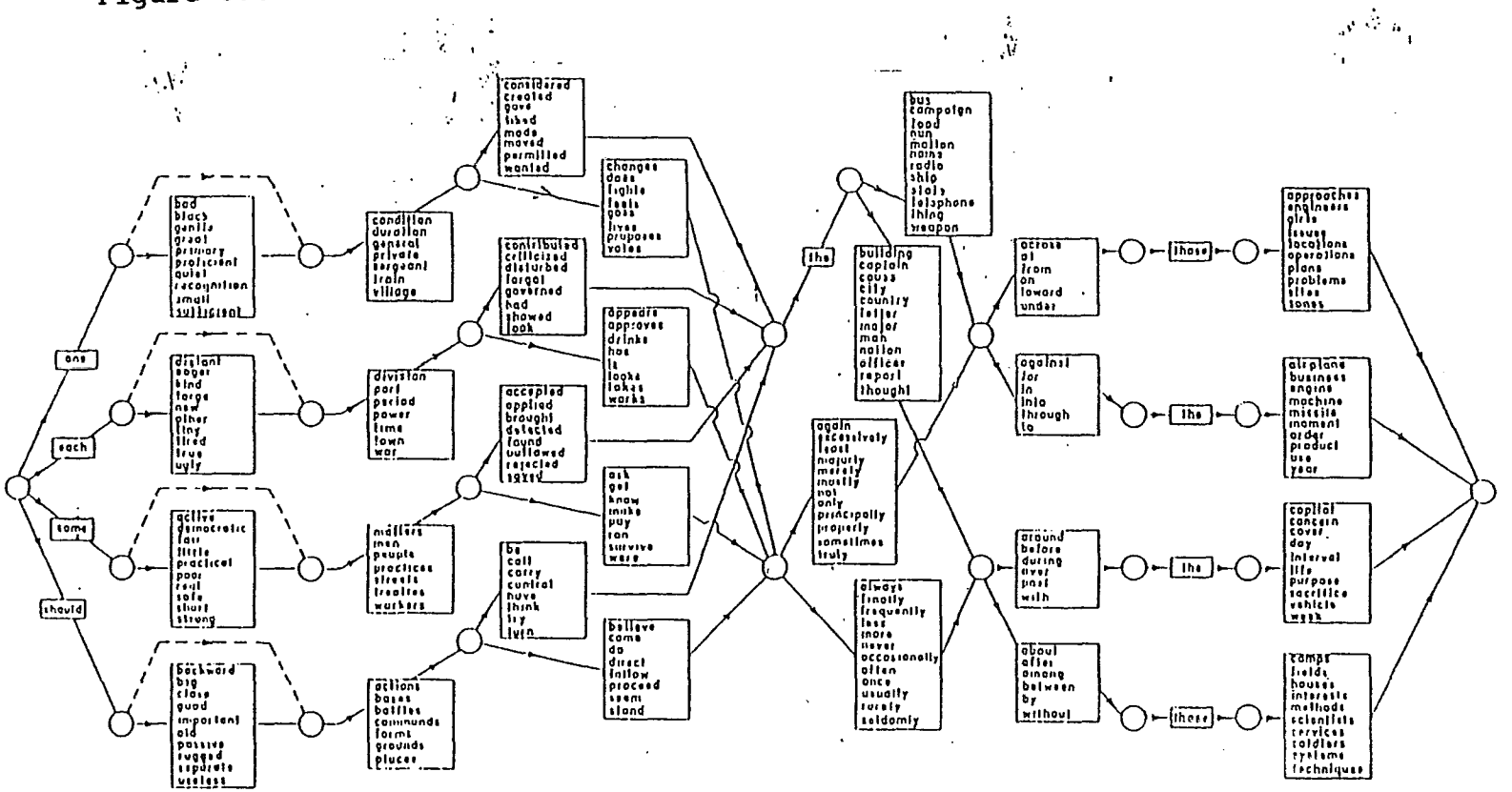
Figure 6.4 is the model of the artificial Raleigh language (Jelinek et al. 1983). For each word, there are

initial fixed trigram transition probabilities, which are stored in a secondary storage device. When CSR system is activated, counter fields added to each word to keep track of its usage frequencies would be allocated into a primary memory for storge efficiency. During the transactions of CSR, the usage frequencies for each word are cumulated. After CSR finishes its transactions, a self-organizing mechanism incorporating count would be activated in order to update $P(w)$.

Suppose that the previous two words are "the" and "bus" (e.g., taken from the part circled in Figure 6.4) and the initial transition probabilities for next words are given in Figure 6.5. After transactions, the frequencies for each word shown in Table 6.2 will be obtained. Then, the updated probabilities can be obtained by multiplying the probabilities $P(w)$ with the weighted frequencies. Since the summation of the updated frequencies is not equal to one, the normalization processes are conducted again, thus providing final updated probabilities $P(w')$. The probabilities $P(w')$ would be stored in the secondary storage and be used for a next transaction. Of course, the counter fields would be deleted from a primary storage after transactions for other computer usages. In summary, this proposed new framework of CSR system incorporating self-

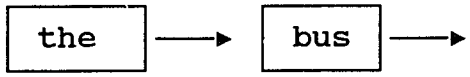
organizing statistical language model adapts to a speaker by attempting to constantly update the probabilities $P(w)$ in the text generator, thus generating texts in a more natural manner.

Figure 6.4: Grammar of the Raleigh language.



Source: Jelinek, F., Mercer, R. L., and Bahl, L. R., "A maximum likelihood approach to continuous speech recognition, " *IEEE Transactions on Patterns Analysis and Machine Intelligence*, Vol. 5-PAMI, No. 2, pp. 179-190, 1983.

Figure 6.5 : The Transition Probabilities before Update.



Words	*P(w)*
across	0.35
at	0.25
from	0.20
on	0.10
forward	0.05
under	0.05

Table 6.2 : The Transition Probabilities after Update.

Words	*P(w)*	FREQ	Updating (a)	*P(w')*
across	0.35	3	0.35 X (3/20) = 0.0525	0.1813
at	0.25	6	0.25 X (6/20) = 0.1620	0.5597
from	0.20	5	0.20 X (5/20) = 0.0500	0.1727
on	0.10	4	0.10 X (4/20) = 0.0200	0.0691
forward	0.05	1	0.05 X (1/20) = 0.0025	0.0086
under	0.05	1	0.05 X (1/20) = 0.0025	0.0086
SUM	1.00	20	(b) 0.2895	1.0000

P(w') = the normalized probability calculated by (a)/(b).

CHAPTER 7

CONCLUSION

Most computer information systems involve the usage-dependent phenomenon and their performance depends on the pattern of the phenomenon. Thus, modeling the usage-dependent phenomenon is significantly important in order to take full advantage of the phenomenon and incorporate it effectively into systems design. A usage process model (the Simon-Yule model) is proposed for modeling the usage-dependent phenomenon. The model is constructive, not subject to many limitations of the other approaches (i.e., a usage index or a usage distribution approach), and simple enough to use by practitioners.

The usage process model is modified and successfully applied to the performance evaluation of the self-organizing linear search heuristics. By applying the model, a more realistic performance evaluation of the heuristics is provided and the open problems which remain unsolved for a long time are solved; i.e., (1) relaxes the previous unrealistic assumptions; (2) derives the theoretical bounds of the heuristics with reasonable assumptions; (3) provides

a reasonable scheme for optimizing the heuristics; and (4) successfully captures the locality phenomenon in the request sequences.

The results obtained from the theoretical and empirical study show that count is the best among the three heuristics being widely used by practitioners. If counter fields are needed for a special purpose, count would be the right choice. Since count takes up storage space for counter fields, however, memoryless heuristics like MTF or transpose are preferred. Based on the results, we suggest the initial use of MTF and the switch to transpose later. A new hybrid rule to providing a more reasonable switching time from MTF to transpose is also proposed. Specifically, if $\alpha \leq 0.4$ and $t \geq 180$, then switch from MTF to transpose. If there is a strong locality tendency in the request sequences, however, MTF is preferred to transpose all the time.

To show how the previous findings can be incorporated into a UDIS design, an adaptive framework incorporating a self-organizing mechanism is proposed for continuous speech recognition systems based on the Simon's theoretical point of view of the text generation (i.e., imitation and association). For a self-organizing mechanism, count is incorporated into a statistical language model for CSR in order to update the transition probabilities $P(w)$

constantly. This proposed adaptive framework for CSR would provide a more reliable statistical model of text for CSR.

One possible extension of this research is to derive more theoretical bounds of other heuristics for realistic comparisons between the heuristics. In addition, a tighter bound of MTF and a closed-form of the relative efficiency of MTF would be desirable. Another possible extension is to apply the usage process model and self-organizing linear search heuristics to other application areas which are identified in Chapter 1. One interesting application would be a usage-dependent menu design. Most current computer information systems are required to provide more user-friendly user interfaces. A menu system is clearly one of the prevalent user interfaces, due to its nature of being easy-to-use and easy-to-learn. As discussed in Chapter 1, however, most current menu systems employ static menu structures, requiring users to adjust to large and complex fixed menu configuration. By incorporating the heuristics into menu systems, we might develop a self-organizing dynamic menu systems so that users can reduce the number of menu selections to access the information they need.

REFERENCES

Anderson, E. J., et al., "A counter example to optimal list ordering," *Journal of Applied Probability*, Vol. 19, No. 3, pp. 730-732, 1982.

Bahl, L. R., et al., "A tree based statistical language model for natural language speech recognition," *IEEE Trans. on Acoustic, Speech, and Signal Processing*, Vol. 37, No. 7, pp. 1001-1008, 1989.

Bayman, P., Civanlar, S., and Whitten, W. B., "Usage-sensitive menu design with Huffman coding," *Proceedings of the 22nd Annual Hawaii International Conference on Systems Science*, pp. 436-437, 1989.

Bellow, M. L, "Autoregressive performance analysis of self-organizing data structures," *Computer Science and Statistics: Proceedings of the 19th Symposium on the Interface*, pp. 417-421,

Bentley, J. L. and McGeoch, C. C., "Amortized analyses of self-organizing sequential search heuristics," *Communications of the ACM*, Vol. 24, No. 4, pp. 404-411, 1985.

Bitner, J. R., "Heuristics that dynamically organize data structures," *SIAM Journal of Computing*, Vol. 8, No. 1, pp. 82-110, 1979.

Booth, A. D., "A law occurrences for words of low frequency," *Information and Control*, Vol. 10, No. 4, pp. 386-393, 1967.

Burville, P. J., and Kingman, E. C., "On a model for storage and search," *Journal of Applied Probability*, Vol. 10, No. 3, pp. 697-701, 1973.

Chen, W. C., "On the weak form of Zipf's law," *Journal of Applied Probability*, Vol. 17, pp. 611-622, 1980.

Chen, Y. S. and Leimkuhler F. F., "Analysis of Zipf's law: An index approach," *Information Processing and Management*, Vol. 23, No. 3, pp. 171-182, 1987.

Chen, Y. S., "An exponential recurrence distribution in the Simon-Yule model of text," *Cybernetics and Systems: An International Journal*, Vol. 19, pp. 521-545, 1988.

Chen, Y. S., "Zipf's laws in text modeling," *International Journal of General Systems*, Vol. 15, pp. 233-252, 1989.

Chen, Y. S. and Leimkuhler F. F., "A type-token identity in the Simon-Yule model of text," *Journal of the American Society for Information Science*, Vol. 40, No. 1, pp. 45-53, 1989.

Chen, Y. S., "Statistical models of text in continuous speech recognition," *Kybernets*, forthcoming-1.

Chen, Y. S., "A type-token study of statistical models of text," *Cybernetica*, forthcoming-2.

Chung, F. R. K., et al., "Self-organizing sequential search and Hilbert's inequalities," *Journal of Computer and Systems Science*, Vol. 36, pp. 148-157, 1988.

Ellis, S. R. and Hitchcock, R. J., "The emergence of Zipf's law: Spontaneous encoding optimization by users of a command language," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-16, No. 3, May/June 1986.

Frederickson, G. N., "Self-organizing heuristics for implicit data structures," *SIAM Journal of Computing*, Vol. 13, No. 2, pp. 277-291, 1984.

Fedorwicz, J., "A Zipfian model of an automatic bibliographic system: an approach to MEDLINE," *Journal of the American Society for Information Science*, pp. 223-232, July 1982.

Fedorwicz, J., "The theoretical foundation of Zipf's law and its application to the bibliographic database environment," *Journal of the American Society for Information Science*, pp. 285-293, September 1982.

_____, "Database performance evaluation in an indexed file environment," *ACM Transactions on Database Systems*, Vol. 12, No. 1, pp. 85-110, March 1987.

Gleser, C. J. and Moore, D. S., "The effect of dependence on Chi-Squared and empiric distribution tests of fit," *The Annals of Statistics*, Vol. 11, No. 4, pp. 1100-1108, 1983.

Gonnet, G. H., et al., "Toward self-organizing linear search," *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science* (San Juan, Puereto Rico, Oct. 1979), IEEE, New York, pp. 169-174.

Good, I. J., *Good Thinking: The Foundations of Probability and Its Applications*, Minneapolis University of Minnesota Press, 1983.

Halstead, M. H., *Elements of Software Science,* Elsevier/North-Holland, New York, 1977.

Hamming, R. W., *Coding and Information Theory*, 2nd ed., Prentice-Hall, 1986.

Heising, W. P., "Note on random addressing techniques," *IBM Systems Journal*, Vol. 2, No. 2, pp. 112-116, June 1953.

Hendricks, W. J., "The stationary distribution of an interesting Markov chain," *Journal of Applied Probability*, Vol. 9, No. 1, pp. 231-233, 1972.

Hendricks, W. J., "An extension of a theorem concerning an interesting Markov chain," *Journal of Applied Probability*, Vol. 10, No. 4, pp. 886-890, 1973.

Hendricks, W. J., "An account of self-organizing systems," *SIAM Journal of Computing*, Vol. 5, No. 4, pp. 715-723, 1976.

Herron, D., "Industrial engineering applications of ABC curves," *AIIE Transactions*, Vol. 8, No. 2, pp. 210-218, 1976.

Hester, J. H., and Hirschberg, D. S., "Self-organizing search lists using probabilistic back-pointers," *Tech Rep. 85-14,* Dept. of Information and Computer Science, Univ. of California, Irvine.

Hester, J. H., and Hirschberg, D. S., "Self-organizing linear search," *Computing Surveys*, Vol. 17, No. 3, pp. 295-311, 1985.

Ijiri, Y., and Simon, H. A., *Skew Distributions and the Sizes of Business Firms*, North-Holland Publishing Co., 1977.

Jelinek, F., Mercer, R. L., and Bahl, L. R., "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Patterns Analysis and Machine Intelligence*, Vol. 5-PAMI, No. 2, pp. 179-190, 1983.

Jelinek, F., "The development of an experimental discrete dictation recognizer," *Proceedings of IEEE,* Vol. 73, No. 11, pp. 1616-1624, 1985.

Kan, Y. C., and Ross, S. M., "Optimal list order under partial memory constraints," *Journal of Applied Probability*, Vol. 17, No. 4, pp. 1004-1015, 1980.

Knuth, D. E., *The Art of Computer Programming, Vol. 3 - Sorting and Searching,* Addison Wesley Co., 1973.

Lancaster, F. W., *Information Retrieval Systems: Characteristics, Testing, and Evaluation*, 2nd Ed., John Wiley and Sons, Inc. 1979.

Mandelbrot, B., "An information theory of statistical structure of language," *Proceedings of the Symposium on Applications of Communications Theory*, (London, September 1952), London: Butterworths, 1953, pp. 486-500.

Markov, A. A., "An example of a statistical investigation of the text of 'Eugen Onegin' illustrating the connection of trials in a chain," *Bulletin de L'Academie Imperiale des Science de St. Petersburg,* Vol. 7, 153, 1913.

McCabe, J., "On serial files with relocatable records," *Operations Research*, Vol. 12, pp. 609-618, 1965.

McKeown, K. R., *Text Generation*, Cambridge University Press, 1985.

Pao, M. L., "Automatic text analysis based on transition phenomena of word occurrences," *Journal of the American Society for Information Science*, Vol. 29, No. 3, pp. 121-124, 1978.

Prather, R. E., "Comparison and extension of theories of Zipf and Halstead," *The Computer Journal*, Vol. 31, No. 3, pp. 248-252, 1988.

Rivest, R., "On self-organizing sequential search heuristics," *Communications of the ACM*, Vol. 19, No. 2, pp. 63-67, 1976.

Salton, G., *Automatic Text Processing*, Addison-Wesley Publishing Co., 1989.

Samson, W. B., and Bendell, A., "Rank order distribution and secondary key indexing," *The Computer Journal*, Vol. 28, No. 3, pp. 309-312, 1985.

Sleater, D. D., and Tarjan, R. E., "Amortized efficiency of list update and paging rules," *Communications of the ACM*, Vol. 28, No. 2, pp. 202-208, 1985.

Shannon, C. E., "Prediction and entropy of printed English," *Bell Syst. Tech. J.*, Vol. 30, pp. 50-64, January 1951.

Shooman, M. L., *Software Engineering: Design, Reliability, and Management*, McGraw-Hill, 1983.

Simon, H. A., "On a class of skew distribution functions," *Biometrika*, Vol. 42, pp. 425-440, 1955.

Simon, H. A., "On judging the plausibility of theories, in B. van Rootselaar and J. F. Staal (eds.)," *Logic, Methodology and Philosophy of Sciences*, Vol. III, Amsterdam: North-Holland, 1968.

Simon, H. A., and Van Wormer, T. A., "Some Monte Carlo estimates of the Yule distribution," *Behavior Science*, Vol. 8, pp. 203-210, 1963.

Rivest, R., "On self-organizing sequential search heuristics," *Communications of the ACM*, Vol. 19, No. 2, pp. 63-67, 1976.

Tague, J., and Nicholls, P., "The maximal value of a Zipf size variable: sampling properties and relationship to other parameters," *Information Processing & Management*, Vol. 23, No. 3 pp. 155-170, 1987.

Tenenbaum, A., "Simulations of dynamic sequential search algorithms," *Communications of the ACM*, Vol. 21, No. 9, pp. 790-791, 1978.

Tenenbaum, A., and Nemes, R. M., "Two spectra of self-organizing sequential search algorithms," *SIAM Journal of Computing*, Vol. 14, No. 3, pp. 557-566, 1982.

White, G. M., "Natural language understanding and speech recognition," *Communications of the ACM*, Vol. 33, No. 8, pp. 72-82, 1990.

Wiederhold, G., *File Organization for Database Design*, McGraw-Hill, Inc., 1987.

Young, S. R., et al., "High level knowledge sources in usable speech recognition systems," *Communications of the ACM*, Vol. 32, No. 2, pp. 183-184, 1989.

Yule, G. U., *A Statistical Study of Vocabulary*, Cambridge, England: Cambridge University Press, 1944.

Zipf, G. K., *Human Behavior and the Principle of Least Effort*. Cambridge, MA: Addison-Wesley, 1949.

VITA

Jin-Soo Kim was born in Chung Yang, Korea on November 21, 1956. He is the third son of Mr. Dong Gi Kim and Mrs. Hae Dong Kim. He received his elementary and high school education in Seoul, graduating from the Senior High School Affiliated to A College of Education at Seoul National University. He received his Bachelor of Economics from Yonsei University in Seoul, Korea in 1982. From December 1981 to July 1983, he worked as a financial analyst at Sam Sung Life Insurance Co., Seoul, Korea. In 1984, He came to the United States and received Master of Business Administration in Management Information Systems from University of Texas at Arlington in summer 1986. After working on the Ph.D program at University of Texas at Arlington for one year, he transferred to the Louisiana State University in Baton Rouge and continued the pursuit of a Doctor of Philosophy with a major in Management Information Systems and a minor in Computer Science.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Jin-Soo Kim

Major Field: Quantitative Business Analysis

Title of Dissertation: Usage-Dependent Information Systems Design

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

November 16, 1990