ADAPTIVE LIST ORGANIZING FOR NON-STATIONARY QUERY DISTRIBUTIONS. PART I: THE MOVE-TO-FRONT RULE

B.J. Oommen

SCS-TR-168, JANUARY 1990

+Partially supported by the Natural Sciences and Engineering Research Council of Canada.

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

Adaptive List Organizing for Non-stationary Query Distributions. Part I: The Move-to-Front Rule

R.S. Valiveti and B.J. Oommen [*]
School Of Computer Science
Carleton University
Ottawa, Canada, K1S 5B6

Abstract

Consider a linear list $\mathcal{R}$ composed of all the records in the set $\{R_1, R_2, \cdots, R_N\}$. At a given instant, the records appear in one of the $N!$ permutations of the N elements. It is well known that in order to arrive at the minimum average cost of accesses, the list $\mathcal{R}$ must be arranged in the decreasing order of probabilities of accessing the elements. Many heuristics to dynamically reorganize the list are known in the literature and include the Move-to-Front (MTF) and the Transposition (TR) rules.

The assumption made in all studies of adaptive linear lists has been that the access probability distribution is time invariant, and that the individual accesses are statistically independent. This paper studies the impact of relaxing the time invariance assumption. Two models of non-stationary query distributions are suggested in this paper. The first approach is based on the idea of switching environments [11] and the second model is completely general and allows the probability distribution vector itself to be a random vector.

The MTF rule is analyzed under both these models. The theoretical results presented in the paper have been experimentally validated.

Keywords:
Linear list, Adaptive data structures, List organizing heuristics, Markov Chains, Convergence.

[*]Both authors partially supported by Natural Sciences and Engineering Research Council Of Canada.

1 Introduction

Consider the linear list $\mathcal{R}$ composed of the elements $\{R_1, R_2, \cdots, R_N\}$. At a given time instant, the linear list contains the records in one of the $N!$ permutations of the N elements. At each time instant, the records are accessed, based on the unknown distribution $\mathcal{S}= [s_1, s_2, \cdots, s_N]^T$, where s_i is the probability of accessing the record R_i. It is easy to prove that the optimal order which minimizes the expected cost of access is the one in which the records are arranged in the decreasing order of the access probabilities $\{s_i\}$.

To render the problem non-trivial, we, of course, assume that the s_i's are unknown. Since a random initial arrangement need not be the best for a given distribution, a lot of research effort has been devoted to the aspect of devising schemes that dynamically update (i.e. permute) the list in such a way that future accesses can possibly be made more efficiently. Two rules that have been exhaustively studied are the Move-to-Front (MTF) [10] and the Transposition Rule (TR) [10]. Many excellent reviews on this subject have appeared in the literature. Our bibliography is by no means complete but the interested reader is referred to the review papers [1,4]. For more recent results involving time invariant query distributions, please refer to [8]-[9].

The analysis approach typically consists of representing the $N!$ states of the linear list as the $N!$ states of a Markov Chain. Let us denote the set of permutations by $\mathbf{\Pi} = \{\pi_i \mid i = 1, 2, \ldots, N!\}$ and denote the steady state probability that the Markov Chain is in the state π_i by $Pr(\pi_i)$. If $\text{Cost}(\pi_i)$ represents the expected retrieval cost, given that the elements are in the permutation specified by π_i, and C_τ the asymptotic expected cost of retrieval as per a reorganization strategy τ, then C_τ can be written down as:

$$C_\tau = \sum_{i=1}^{N!} Pr(\pi_i) \cdot \text{Cost}(\pi_i). \qquad (1)$$

Since the stationary probabilities for the MTF and the TR rules are analytically obtainable [2,10], the closed form expressions for the asymptotic expected cost can easily be obtained.

It must be noted that the above analysis schemes rely on the *convergence* of the underlying *Homogeneous* Markov Chain. The convergence properties of the underlying chains can be inferred only if the transition probabilities are independent of time. When the transition probabilities vary with time, predicting the convergence is an extremely difficult problem and the literature available is scanty.

Figure 1 shows the Markov Chain for the case when the linear list being accessed has 3 elements and the MTF heuristic is used to reorganize the list. If $\mathcal{S}$, the vector representing

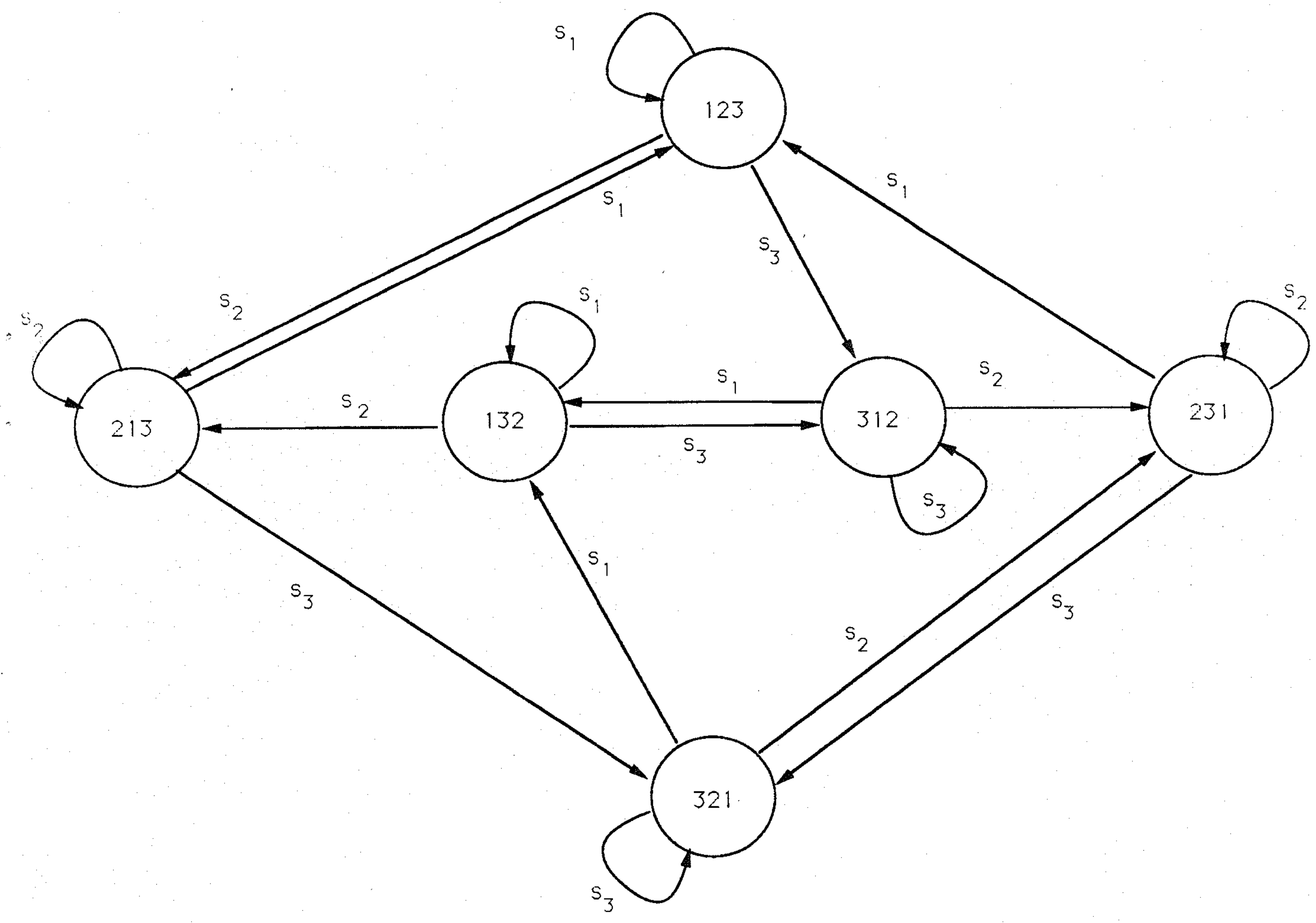

Figure 1: The Markov Chain for the list with 3 elements. The MTF heuristic is used to reorganize the list with every access.

the access probabilities is time varying, the transition probabilities of the Markov Chains are also time varying, and hence even the question of the existence of a stationary distribution is unsolved. Furthermore, even if the stationary distribution for such a chain can be shown to exist, the technique for evaluating it is unknown except for some extremely trivial or straightforward cases.

The only reported list organizing strategy in which the underlying Markov chain is non-homogeneous is the one due to Oommen and Hansen [7]. Their scheme is a variation of the well known MTF rule. In the scheme reported in [7], the accessed element R_i is moved to the front of the list with a probability $f(n) = a^n$ (where $a < 1$) at the time instant n. In other words, $f(n)$ is progressively decreased so that the chain is finally absorbing. The underlying non-homogeneous Markov Chain has the property that the transition matrix can be decomposed into the sum of two matrices, from each of which the term a^n can be factored out. This, augmented with the fact that both these matrices have the same diagonalizing matrix, leads to the solution of the asymptotic distribution. Although these computations are both messy and involved, the techniques used in [7] are novel primarily because it represents one of the few cases in which non-homogeneous chains can be solved explicitly. The important aspect to note is that the underlying chain associated with the *algorithm* is non-homogeneous, even though the user's query distribution is time invariant. In other words, the non-homogeneity is not an inherent property of the underlying *problem.* The non-homogenuity has been introduced *merely because of the heuristic* which is stochastic, and alters its behaviour with time.

This paper endeavours to tackle the problem in which the user's query distribution itself is unknown and yet non-stationary. That is, in this paper we shall consider the effect of having the access probabilities **themselves** change with time. The implications of this are two-fold. First of all, in this case, the chain is non-homogeneous because of the inherent property of the problem solved and not a consequence of the heuristic used. The second implication is that although the non-stationarity may be a consequence of the problem on hand, by a suitable transformation in the solution space, this solution can be obtained by the analysis of an underlying *homogeneous* Markov chain. Both these concepts will be clarified presently.

To grasp the above claims, consider the case in which the query patterns follow one distribution for a random period of time and then switch to another distribution for another random interval. It is clearly possible that these distributions could have the property that the best list organizations for these random distributions are different. We know that adaptive list organizing strategies are very effective when dealing with stationary query distributions. In this connection we would like to examine the effectiveness of known heuristics like MTF and TR, when used in conjunction with such non-stationary query distributions.

Before we tackle this problem, we first have to define the concept of "non-stationarity" within the context of this paper. In other words, we need to restrict the types of time variation that are allowed in the distributions. We shall present two models for query generators whose query distributions are non-stationary. In the first model, we shall consider the query generator to be "state-driven". That is, the query generator itself will be considered to be making transitions within the states of a finite Markov chain. In this model, the distribution vector used by the query generator is dependent only on the current state of the "machine". We shall call this model the Markovian Switching Query Generator (MSQG) and it is analogous to the switching environments, as described in [6,11]. This case is analyzed in Section 2, for the case when the list organizing rule is MTF.

In Section 3, in a more daring attempt to tackle the problem, we make the model for the query generator to be far more general. In this case, we permit the underlying distribution $\mathcal{S}(n)$, at the time instant n to be a random vector in itself, and this vector is drawn from some unknown underlying distribution. This model is called the Random Query Generation (RQG) Model.

In this paper, we shall analyze the MTF rule for the MSQG and RQG models. The techniques that are used for solving these two problems are distinct. In the MSQG model, one possible solution is to transform the non-homogeneous chain into a homogeneous Markov chain based on a product space associated with the states of the query generator and the permutations of the list. Although this approach is theoretically tractable, and would lead to a solution analogous to the one in [11], in reality, the problem is intractable because of the magnanimity of the product space. The technique used for the RQG model will also be explained in greater detail in the appropriate sections.

We also present experimental results which support our analytical derivations. We are currently investigating the Transposition rule, under both these models of query generation.

Throughout this paper, we shall use "{}" to represent sets, and "[]" to represent vectors. Tuples, permutations are ordered sequences will be represented using "<>" with an endeavor to render a relatively complicated notation consistent.

2 Markovian Switching Query Generators

In this section we shall present our first model for non-stationary query distributions. In this approach, we associate the set of states $\{Q_i | 1 \leq i \leq e\}$ with the non-stationary query generator. Furthermore, when the query generator is in the state Q_i, it uses the probability distribution $\mathcal{S}_i = [s_{i1}, s_{i2}, \ldots, s_{iN}]^T$, for generating queries. The components of $\mathcal{S}_i$ obey the following constraint for all $1 \leq i \leq e$:

$$\begin{aligned} 0 \leq \ & s_{ik} \leq 1, \quad \text{for} \quad 1 \leq k \leq N. \\ \sum_{k=1}^{N} & s_{ik} = 1. \end{aligned}$$

We shall assume that the query generator makes transitions between the set of states $\{Q_1, Q_2, \ldots, Q_e\}$ in a Markovian fashion and hence it will itself be modeled as a Markov Chain. As mentioned earlier, although this modeling approach is directly based on the work of Tsetlin in [11], the technique by which we endeavour to solve the problem is distinct from his.

The situation described above corresponds to one of cascading two Markov chains. The query generator forms the first Markov Chain and the set of permutations are the states of the second Markov Chain. The outputs from the first chain effect the transitions made in the second chain. Two approaches to the analysis of cascaded chains can be seen. To present the problem in the correct perspective, we start with the intuitive approach to analyze the cascading of these two chains, which unfortunately is not valid.

In the "intuitive" approach to solve the above problem, we try to solve each of the Markov chains separately, as follows. The first step involves calculating the equilibrium probabilities of the first chain (i.e. the one associated with the query generator). Based on these, we calculate the steady state query distribution $\bar{S}$ which is observable as the "output" of this chain obtained using the law of total probability. With this information about $\bar{S}$ at hand, the steady state probabilities of the list being in one of its $N!$ arrangements can be easily written down, when the reorganizing heuristic is either the MTF or TR. Although the technique is easy to comprehend, the approach is fallacious because of the following argument. We note that output distribution of the first chain is changing with time. Moreover, the results for the equilibrium probabilities for the Markov chain representing the MTF rule can be used only if the input distribution **did not change with time**. Thus before this chain can converge to the equilibrium probabilities for a given distribution, it is faced with a new input distribution. In summary, this cascading approach is invalid for general Markov chains. The problems are similar to those encountered in [11] in which the equilibrium distribution is shown to be function not only of $\bar{S}$ but also of the mean time between switches.

2.1 Exact Analysis Using the Composite Chain

2.1.1 Asymptotic Analysis for the Query Generation System

An exact analysis of the MTF rule would require that we construct a composite Markov chain, in which the resultant state space is the cross product of the states of the list, and the states of the query generator. The set of states of the list is simply the set of all permutations $\Pi = \{\pi_i \mid i = 1, 2, \ldots, N!\}$. The query generator $\mathcal{Q}$ can be assumed to be in one of the states of the set $\{Q_1, Q_2, \cdots, Q_e\}$. We shall denote by $Q(n)$ the state of the query generator at the time instant n. The transition matrix for the Markov Chain representing the query generator is Δ, where,

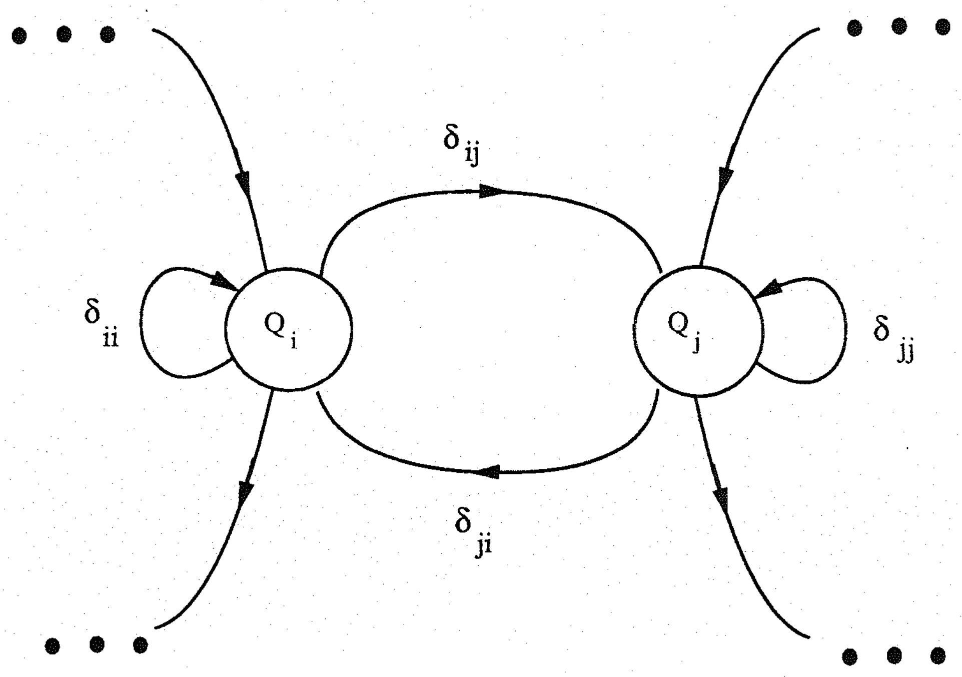

Figure 2: This figure shows a few typical transitions made by the query generator. Only two representative states of the query generator, i.e. Q_i, and Q_j are shown. The labels beside the transition arcs denote transition probabilities.

$$\Delta = \begin{bmatrix} \delta_{11} & \cdots & \delta_{1e} \\ \delta_{21} & \cdots & \delta_{2e} \\ \vdots & \ddots & \vdots \\ \delta_{e1} & \cdots & \delta_{ee} \end{bmatrix}, \tag{2}$$

where, $\delta_{ij} = Pr(Q(n+1) = Q_j \mid Q(n) = Q_i)$. The transition map of the MSQG is shown in Figure 2.

As mentioned earlier, when the query generator is in the state Q_i, it uses the probability distribution $\mathcal{S}_i$ to generate accesses to the elements of the linear list, where $\mathcal{S}_i = [s_{i1}, s_{i2}, \ldots, s_{iN}]^T$.

Throughout this paper, we assume that the Markov Chain of the MSQG is ergodic. An understanding of the concepts introduced here leads to a trivial solution in the case when the chain is absorbing. Since the chain will be assumed to be ergodic, it will possess a unique stationary distribution, which is independent of the initial state occupancy probabilities. We shall denote the state occupancy probability vector at the time instant n by $\mathbf{P}(n) = [p_1(n), p_2(n), \ldots, p_e(n)]^T$ and the asymptotic value of $\mathbf{P}(n)$ by $\mathbf{P}^* = [p_1^*, p_2^*, \ldots, p_e^*]^T$ which satisfies the following equation [6]:

$$\Delta^T \cdot \mathbf{P}^* = \mathbf{P}^*. \tag{3}$$

It is well known that the eigenvalues of the matrix defined by (2) determine the rate of convergence of the probability $\mathbf{P}(n)$. Let us denote the set of eigenvalues of Δ by $\{\lambda_1, \lambda_2, \ldots, \lambda_e\}$

and let these be in descending order of their magnitudes. Clearly, $\lambda_1 = 1$ since unity must be an eigenvalue of every stochastic matrix. The eigenvectors of Δ are given by $\{\underline{v}_m\}$, where $\underline{v}_m$ corresponds to the eigenvalue λ_m. It is shown in [6] that closed form expressions for $\mathbf{P}(n)$ can be written down as:

$$\mathbf{P}(n) = \sum_{m=1}^{e} a_m \lambda_m^n \underline{v}_m. \tag{4}$$

where, a_m's are constants depending on $\mathbf{P}(0)$. Additionally, we know that $\underline{v}_1 = \mathbf{P}^*$, since $\lambda_1 = a_1 = 1$.

We can use the expression in (4) to analyze the behaviour of the MSQG as a function of time. Now, the probability of accessing the record R_i at the time instant n can be written down as:

$$\begin{aligned} s_i(n) &= Pr(R_i \text{ is accessed at time } n) \\ &= \textstyle\sum_{m=1}^{e} Pr(R_i \text{ is accessed} \mid Q(n) = Q_m) \cdot Pr(Q(n) = Q_m) \\ &= \sum_{m=1}^{e} s_{mi} \cdot Pr(Q(n) = Q_m) \\ &= \sum_{m=1}^{e} s_{mi} \cdot p_m(n). \end{aligned} \tag{5}$$

Using (4), it is easy to see that $s_i(n)$ can be expressed in the form:

$$s_i(n) = \bar{s}_i + \sum_{m=2}^{e} c_m \lambda_m^n. \tag{6}$$

where, $\bar{s}_i = \sum_{l=1}^{e} s_{li} \cdot p_l^*$ and is the asymptotic value of $s_i(n)$.

With this insight, the asymptotic expected query distribution (when the MSQG has converged) is obtained using the law of total probability as:

$$\bar{\mathcal{S}} = \sum_{i=1}^{e} p_i^* \mathcal{S}_i. \tag{7}$$

This implies that the j^{th} component of $\bar{\mathcal{S}}$ is the weighted combination of the j^{th} components of the vectors $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_e\}$.

2.1.2 Analysis of the Composite Chain

Having defined the characteristics of the query generator, we now turn our attention to the composite chain mentioned in the preamble of Section 2.1.1. Each state in the composite Markov chain is an ordered pair of the form $< \pi_i, Q_m >$. Let us denote the state of the composite Markov Chain at the time instant n by $\phi(n)$. Then $\phi(n)$ is simply the ordered pair $< L(n), Q(n) >$, where $L(n)$ and $Q(n)$ represent the state of the list and the query generator at time instant n. The transition probabilities of the composite Markov Chain are derived according to the equation:

$$\begin{aligned} & Pr(\phi(n+1) =< \pi_i, Q_m >| \phi(n) =< \pi_k, Q_l >) \\ &= Pr(L(n+1) = \pi_i, Q(n+1) = Q_m \mid \phi(n) =< \pi_k, Q_l >), \qquad (8) \\ &= Pr(L(n+1) = \pi_i \mid \phi(n) =< \pi_k, Q_l >, Q(n+1) = Q_m) \\ &\quad \cdot Pr(Q(n+1) = Q_m \mid \phi(n) =< \pi_k, Q_l >) \qquad \text{(using the chain rule)} \end{aligned}$$

which reduces to the following:

$$\begin{aligned} &= Pr(L(n+1) = \pi_i \mid L(n) = \pi_k, Q(n) = Q_l) \cdot Pr(Q(n+1) = Q_m \mid Q(n) = Q_l) \\ &\qquad \text{(dropping extraneous conditions)} \\ &= Pr(L(n+1) = \pi_i \mid L(n) = \pi_k, Q(n) = Q_l) \cdot \delta_{lm}. \qquad (9) \end{aligned}$$

For the general case, when the list to be ordered has N elements, and the environment has e states, the composite chain has $e \cdot N!$ states. If the resulting Markov Chain is ergodic, the problem is "simply" one of computing *its* stationary distribution. Let $Pr^*(< \pi_i, Q_m >)$ be the equilibrium (stationary) probability of the composite chain being in the state $< \pi_i, Q_m >$. Once these probabilities are known, the expected retrieval cost can be easily computed from the following generalization of (1):

$$C_\tau = \sum_{m=1}^{e} \sum_{i=1}^{N!} Pr^*(< \pi_i, Q_m >) \cdot \text{Cost}(\pi_i, Q_m). \qquad (10)$$

where $\text{Cost}(\pi_i, Q_m)$ is the expected cost associated with the permutation π_i if it was interacting with the environment Q_m.

Although (10) is the exact formula for computing the cost, its direct use is, of course, cumbersome, due to the large number ($e \cdot N!$) of stationary probabilities that must be computed. Thus even though this approach is analytically tractable, we would prefer an alternate method to compute C_τ. It can be shown that the expected cost for *any* linear list organizing heuristic can be written down as [10]:

$$C_\tau = \sum_{j=1}^{N} s_j \left\{ \sum_{\substack{i=1,2,\ldots,N \\ i \neq j}} (1 + b(i,j)) \right\}. \qquad (11)$$

where s_j denotes the probability of accessing element R_j and $b(i,j)$ represents the probability that the element R_i precedes the element R_j. For any given environment, (10) and (11) are equivalent, and since the $b(i,j)$'s are more easily computable, throughout this paper (as in all the current literature) we opt to compute C_τ using (11) as opposed to (10).

We shall now present our first result for the MSQG system in which $N = 2$ and the number of environments is arbitrary.

Theorem 1 *Consider the case of MSQG in which the list being organized has two elements ($N = 2$) and the query generator can be in one of e states, governed by the transition matrix defined in (2). Then, $Pr^*(<ij>)$, the asymptotic probability that the list is in the state $<ij>$, is given by:*

$$Pr^*(<ij>) = \sum_{m=1}^{e} p_m^* s_{mi}.$$

Proof:

Let $w_{ij;m}(n)$ denote the probability that the composite chain is in the state $<ij; Q_m>$, at the time instant n, where $i, j = 1, 2$ and $i \neq j$. Then, using (9), the transition probabilities in the composite Markov chain can be written down as:

$$Pr(\phi(n+1) =< ij; Q_m > | \ \phi(n) =< uv; Q_w >) = s_{wi}\delta_{wm}.$$

In this scenario, the typical transitions in the composite Markov Chain are shown in Figure 3. From an examination of the composite Markov chain in Figure 3 we can write the following equations:

$$\begin{aligned} w_{ij;m}(n+1) &= \sum_{l=1}^{e}\{w_{ij;l}(n)s_{li}\delta_{lm}\} + \sum_{l=1}^{e}\{w_{ji;l}(n)s_{li}\delta_{lm}\} \\ &= \sum_{l=1}^{e} s_{li}\{w_{ij;l}(n) + w_{ji;l}(n)\}\delta_{lm}. \end{aligned} \qquad (12)$$

Using (12), and the laws of total probability, we can compute the total probability that the list will be in the order $<ij>$ at the time instant $n+1$ as:

$$\begin{aligned} Pr(L(n+1) =< ij >) &= \sum_{m=1}^{e} w_{ij;m}(n+1) \\ &= \sum_{m=1}^{e}\sum_{l=1}^{e} s_{li}\{w_{ij;l}(n) + w_{ji;l}(n)\}\delta_{lm} \end{aligned}$$

(by interchanging the order of summations)

$$\begin{aligned} Pr(L(n+1) =< ij >) &= \sum_{l=1}^{e}\sum_{m=1}^{e} s_{li}\{w_{ij;l}(n) + w_{ji;l}(n)\}\delta_{lm} \\ &= \sum_{l=1}^{e} s_{li}\{w_{ij;l}(n) + w_{ji;l}(n)\}\left(\sum_{m=1}^{e}\delta_{lm}\right) \end{aligned}$$

and since $\sum_{m=1}^{e}\delta_{lm} = 1$, we get

$$Pr(L(n+1) =< ij >) = \sum_{l=1}^{e} s_{li}\{w_{ij;l}(n) + w_{ji;l}(n)\} \qquad (13)$$

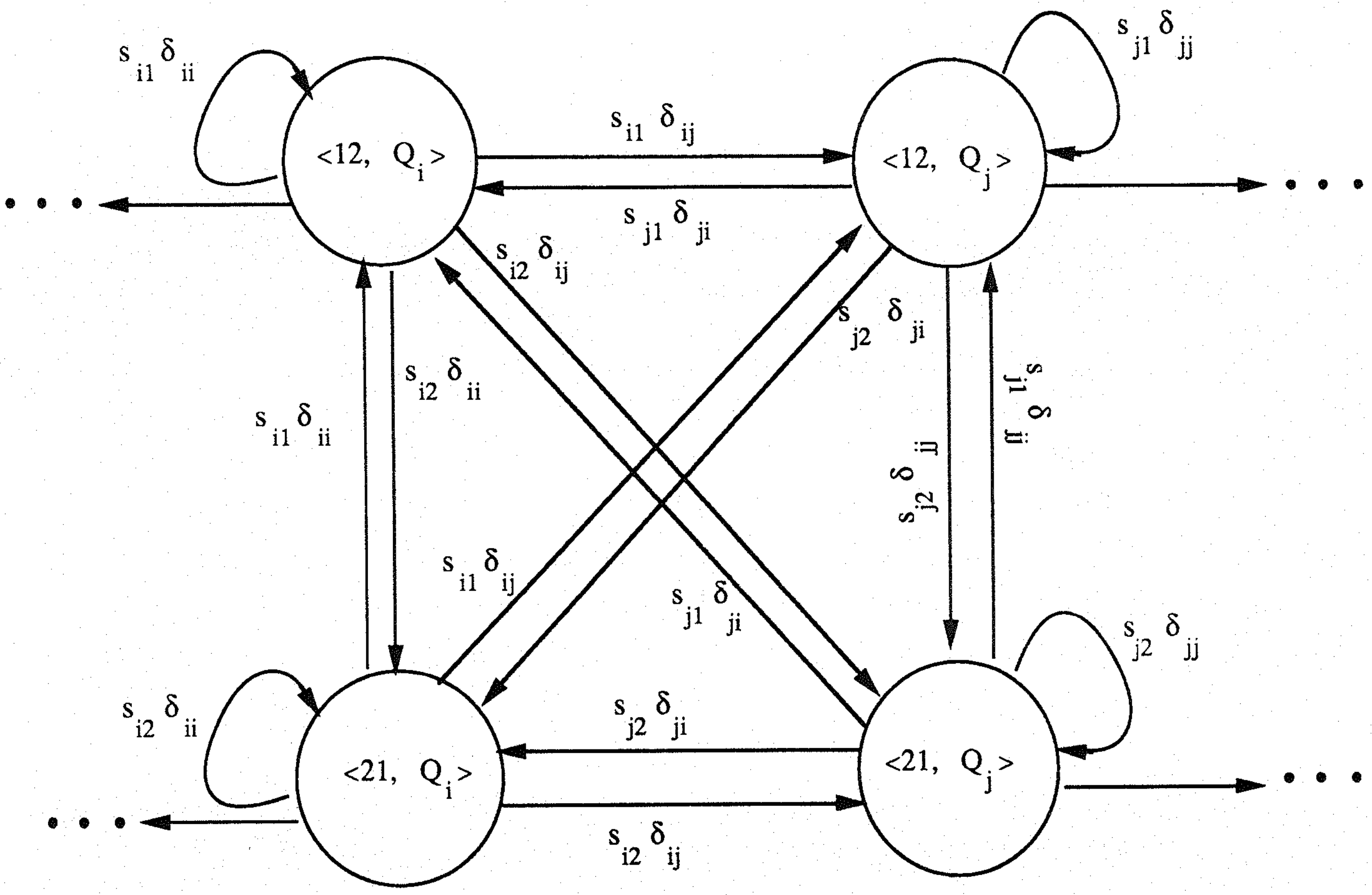

Figure 3: This figure depicts typical transitions in the composite Markov chain for the system in which the list $\mathcal{R}$ has 2 elements. For the sake of illustration, the query generator switches between two states i and j (see Figure 2).

Consider the term $w_{ij;l}(n) + w_{ji;l}(n)$. Since this term is independent of the ordering of the elements of the list, and only involves l, the index of the environment, we have,

$$w_{ij;l}(n) + w_{ji;l}(n) = Pr(Q(n) = Q_l). \tag{14}$$

Using (14), (13) can be rewritten as:

$$Pr(L(n+1) =< ij >) \quad = \quad \sum_{l=1}^{e} s_{li} Pr(Q(n) = Q_l) \tag{15}$$

Since the Markov chain representing the query generator be ergodic, $\lim_{n\to\infty} Pr(Q(n) = Q_l)$ exists as given by (3). Thus, we conclude that,

$$Pr^*(< ij >) = \lim_{n\to\infty} Pr(L(n) =< ij >) = \sum_{l=1}^{e} p_l^* s_{li}.$$

which proves the result. □

Even though the number of elements N is just 2, the computation of the asymptotic probabilities of the composite chain is quite involved. It is coincidental that even though this computation is elaborate, it can be simplified using (14). However the insight we get from this analysis is interesting because it presents us with a technique by which the asymptotic probabilities of the individual chains can be coalesced to give us the asymptotic properties of the composite chain. Interestingly , the *actual* elements of Δ, the transition probability matrix of the MSQG, do not explicitly appear in the asymptotic probabilities; they are implicitly included by way of the asymptotic probabilities $\{p_i^*\}$.

It is clear that the above techniques can be used if the number of records is $N > 2$. In this case, as mentioned earlier, the composite Markov chain has $e \cdot N!$ states and evaluating the eigenvectors of the $(e \cdot N!) \times (e \cdot N!)$ matrix, the asymptotic probabilities of the resulting Markov Chain can be derived. However this strategy is ludicrous even for values of $N \geq 4$, and so we will have to search for an alternate strategy. In order to do so, rather than concentrate on the entire list, we will focus our attention on the asymptotic probability of an element R_i preceding another element R_j. This, of course, is quite in step with the traditional concepts involving stationary environments.

We now state and prove a general result concerning the asymptotic probability of record R_i preceding R_j in the case of the MSQG model, when the number of records N is arbitrary.

Theorem 2 *Let ${}_iP_j(n)$ denote the probability that record R_i precedes R_j at the time instant n, whose asymptotic value is ${}_iP_j^*$. Then, if $\bar{S}$ is the mean asymptotic query distribution defined by (7), the following results hold:*

1. $${}_iP_j(n+1) = \left(\prod_{k=1}^{n} \{1 - s_i(k-1) - s_j(k-1)\}\right) \cdot {}_iP_j(0) + \\ + \sum_{k=1}^{n} s_i(k-1) \left(\prod_{l=k+1}^{n} \{1 - s_i(l-1) - s_j(l-1)\}\right).$$
2. $${}_iP_j^* = \lim_{n\to\infty} {}_iP_j(n) = \frac{\bar{s}_i}{\bar{s}_i + \bar{s}_j}.$$

Proof:

By the nature of the heuristic used, we can write down the recursive form for the distribution of the *total* probabilities ${}_iP_j(n+1)$ as:

$$ {}_iP_j(n+1) = \begin{cases} 1 & \text{if record } R_i \text{ is accessed at time } n \\ 0 & \text{if record } R_j \text{ is accessed at time } n \\ {}_iP_j(n) & \text{if some record other than } R_i \text{ or } R_j \\ & \text{was accessed at time } n \end{cases} \tag{16}$$

We note that the record R_i is accessed at the time instant n, with probability $s_i(n)$. Also the probability that some record other then R_i or R_j is accessed at time instant n is simply given by $1 - s_i(n) - s_j(n)$. The probabilities mentioned above can be rewritten in the form[1]:

$$ {}_iP_j(n+1) = \begin{cases} 1 & \text{w.p. } s_i(n) \\ 0 & \text{w.p. } s_j(n) \\ {}_iP_j(n) & \text{w.p. } 1 - s_i(n) - s_j(n). \end{cases} \tag{17}$$

Hence, we can write down the total probability ${}_iP_j(n+1)$ as follows:

$$\begin{aligned} {}_iP_j(n+1) &= 1 \cdot s_i(n) + 0 \cdot s_j(n) + {}_iP_j(n) \cdot (1 - s_i(n) - s_j(n)) \\ &= s_i(n) + {}_iP_j(n) \cdot (1 - s_i(n) - s_j(n)). \end{aligned} \tag{18}$$

This is a difference (or recurrence) equation, whose solution yields the required probability of the form ${}_iP_j(n)$[2]. Since the coefficients are not constants, the difference equation is not trivially solved and we shall tackle this presently.

For the ease of the derivation, we now introduce some simplifying notation:

$$\begin{aligned} x_n &\triangleq {}_iP_j(n) \\ b_{n+1} &\triangleq 1 - s_i(n) - s_j(n) \\ c_{n+1} &\triangleq s_i(n). \end{aligned}$$

With this notation, the difference equation (18) can be rewritten as:

[1] w.p. denotes "with probability"

[2] If the query distribution were to be stationary, we would have $s_i(n) = s_i$ for all n. In this case, the difference equation simply reduces to one with constant coefficients. The solution method is straightforward, and it can be easily shown that for this special case, $\lim_{n\to\infty} {}_iP_j(n) = s_i/(s_i + s_j)$. This clearly agrees with the result derived in [10].

$$x_{n+1} = b_{n+1}x_n + c_{n+1}. \tag{19}$$

Consider the following transformation of variables.

$$\begin{aligned} x_0 &= y_0 \\ x_n &= \left(\prod_{k=1}^{n} b_k\right) \cdot y_n \text{ for } n > 0 \end{aligned} \tag{20}$$

Using this substitution, (19) becomes:

$$\left(\prod_{k=1}^{n+1} b_k\right) \cdot y_{n+1} = b_{n+1} \cdot \left(\prod_{k=1}^{n} b_k\right) \cdot y_n + c_{n+1}.$$

If we divide both sides of this equation by $\left(\prod_{k=1}^{n+1} b_k\right)$, we get:

$$y_{n+1} = y_n + d_{n+1} \tag{21}$$

where

$$d_{n+1} = c_{n+1} / \prod_{k=1}^{n+1} b_k$$

It is now obvious that the difference equation (21) has the simple solution:

$$y_n = y_0 + \sum_{j=1}^{n} d_j \tag{22}$$

Hence, by transforming the solution in (22) back to the original variables of our problem, we get the general solution of the original difference equation (18) as:

$$\begin{aligned} x_n &= \left(\prod_{k=1}^{n} b_k\right) \cdot x_0 + \left(\prod_{k=1}^{n} b_k\right) \cdot \left\{\sum_{l=1}^{n} \frac{c_l}{\prod_{k=1}^{l} b_k}\right\} \\ &= \left(\prod_{k=1}^{n} b_k\right) \cdot x_0 + \sum_{k=1}^{n} c_k \left(\prod_{l=k+1}^{n} b_l\right). \end{aligned} \tag{23}$$

In terms of the original variables of our problem, we rewrite (23) as:

$$\begin{aligned} {}_iP_j(n+1) = {} & \left(\prod_{k=1}^{n} \{1 - s_i(k-1) - s_j(k-1)\}\right) \cdot {}_iP_j(0) + \\ & + \sum_{k=1}^{n} s_i(k-1) \left(\prod_{l=k+1}^{n} \{1 - s_i(l-1) - s_j(l-1)\}\right). \end{aligned} \tag{24}$$

and the first result of the theorem is proved.

Although (24) gives the general expression for the probability that a record R_i precedes another record R_j at a given instant n, it is inconvenient to use this to derive the asymptotic value of the probability. In fact, we first need to prove the *existence* of the asymptotic value. We start with the expression for $s_i(n)$ as given by (6). It follows very simply that:

$$\begin{aligned} s_i(n) &= \bar{s}_i + o(\lambda_2^n) \\ \text{and} & \\ s_j(n) &= \bar{s}_j + o(\lambda_2^n) \end{aligned}$$

Since the Markov chain for the MSQG is ergodic, we know that $|\lambda_2| < 1$, and hence $\lambda_2^n \to 0$, as $n \to \infty$. Because of this property, it possible to choose a sufficiently large number, n_0 such that $\forall n > n_0$, $o(\lambda_2^n) \ll \min(\bar{s}_i, \bar{s}_j)$. It is also clear that any $n_0' > n_0$ also possesses this property. Our aim now is to find upper and lower bounds on the values of ${}_iP_j(n)$. These bounds can assist in the determination of the asymptotic value of ${}_iP_j(n)$. Now,

$$\begin{aligned} s_i(n) &= \bar{s}_i + \sum_{m=2}^{e} c_m \lambda_m^n \\ &\leq \bar{s}_i + e|c_{max}| \cdot |\lambda_2|^{n_0} \quad \forall n > n_0. \end{aligned} \tag{25}$$

Similarly,

$$s_i(n) \geq \bar{s}_i - e|c_{max}| \cdot |\lambda_2|^{n_0} \quad \forall n > n_0. \tag{26}$$

Using (25) and (26), we can bound the quantity $s_i(n) + s_j(n)$ as:

$$\bar{s}_i + \bar{s}_j - 2e|c_{max}| \cdot |\lambda_2|^{n_0} \leq s_i(n) + s_j(n) \leq \bar{s}_i + \bar{s}_j + 2e|c_{max}| \cdot |\lambda_2|^{n_0} \quad \forall n > n_0.$$

Or alternately,

$$1 - \bar{s}_i - \bar{s}_j - 2e|c_{max}| \cdot |\lambda_2|^{n_0} \leq 1 - s_i(n) - s_j(n) \leq 1 - \bar{s}_i - \bar{s}_j + 2e|c_{max}| \cdot |\lambda_2|^{n_0} \quad \forall n > n_0.$$

Consider the two new difference equations involving the quantities ${}_i\bar{P}_j(n)$ and ${}_i\underline{P}_j(n)$, which are intended to represent an upper bound and lower bound on ${}_iP_j$ respectively. These difference equations are defined only when $n > n_0$, with $c = e \cdot |c_{max}|$.

$${}_i\bar{P}_j(n+1) = (\bar{s}_i + c|\lambda_2|^{n_0}) + (1 - \bar{s}_i - \bar{s}_j + 2c|\lambda_2|^{n_0}) \cdot {}_i\bar{P}_j(n) \tag{27}$$
$${}_i\underline{P}_j(n+1) = (\bar{s}_i - c|\lambda_2|^{n_0}) + (1 - \bar{s}_i - \bar{s}_j - 2c|\lambda_2|^{n_0}) \cdot {}_i\underline{P}_j(n) \tag{28}$$

It is easy to verify that

$${}_i\bar{P}_j(n) \geq {}_iP_j(n) \geq {}_i\underline{P}_j(n) \qquad \text{for all } n \geq n_0. \tag{29}$$

Now (27) and (28) are constant coefficient difference equations, with asymptotic values given by:

$$\lim_{n\to\infty} {}_i\bar{P}_j(n) = \frac{\bar{s}_i + c|\lambda_2|^{n_0}}{\bar{s}_i + \bar{s}_j - 2c|\lambda_2|^{n_0}} \quad (30)$$

$$\lim_{n\to\infty} {}_i\underline{P}_j(n) = \frac{\bar{s}_i - c|\lambda_2|^{n_0}}{\bar{s}_i + \bar{s}_j + 2c|\lambda_2|^{n_0}} \quad (31)$$

Note that ${}_i\bar{P}_j(n)$ and ${}_i\underline{P}_j(n)$ bound ${}_iP_j(n)$ from above and below respectively. Moreover, we have derived the asymptotic values of ${}_i\bar{P}_j(n)$ and ${}_i\underline{P}_j(n)$ as $n \to \infty$. Hence *if* the quantity ${}_iP_j(n)$ converged as $n \to \infty$, its limit must be bounded by the asymptotic values of ${}_i\bar{P}_j$ and ${}_i\underline{P}_j$. These asymptotic values clearly depend on the choice of n_0. Since the choice of n_0 is arbitrary, we can choose it to be as large as we please. An examination of (30) and (31) reveals that as $n_0 \to \infty$, both these bounds converge to a **common** value, which, by (29) must also be the limit of ${}_iP_j(n)$ as $n \to \infty$. Hence we have established that:

$${}_iP_j^* = \frac{\bar{s}_i}{\bar{s}_i + \bar{s}_j}.$$

which is the second claim of the theorem. □

Remark:

It must be noticed that the result derived in Theorem 1 is a special case of Theorem 2, when the list has exactly two elements (i.e. $N = 2$). That the expressions given by Theorem 1 and Theorem 2 are identical, can be seen by observing that $\bar{s}_i + \bar{s}_j = 1$, if the list has exactly two elements. Since the MTF and TR reorganize the list in the same manner if $N = 2$, the result derived in Theorem 1 applies to the TR heuristic as well. However, if $N > 2$, Theorem 2 only applies to MTF.

3 Random Query Generator

In the previous section, we had demonstrated the analysis technique in the case when the query generator was itself modeled as a Markov chain. In other words, we analyzed the MTF rule in the case when the probability distribution vector $\mathcal{S}$ switched between a set of e vectors. Under this model, the asymptotic value of the probability ${}_iP_j$ was derived.

In this section, we explore generalizations of the above model of query generation. That is, we assume no knowledge of the mechanism by which the query generator is selecting its distributions. The query generation is completely arbitrary and is described by a random *process*. In particular, we no longer require that the probability distributions selected by the query generator be restricted to a finite set and thus, we allow the probability distribution vector $\mathcal{S}(n) = [s_1(n), s_2(n), \ldots, s_N(n)]^T$ to itself be a *random* vector, possessing a distribution $f(\mathcal{S}(n))$. The mean of $\mathcal{S}(n)$ will be denoted by $\boldsymbol{\mu}(n) = [\mu_1(n), \mu_2(n), \ldots, \mu_N(n)]^T$.

A good deal of insight into this model of query generation can be obtained from the following explanation. Let us consider the computation of the *total probability* that an element R_i of the list is accessed. Now,

$$Pr(R_i \text{ is accessed at time } n | \mathcal{S}(n)) = s_i(n). \qquad (32)$$

Hence, taking expectations of both sides of (32), we obtain $Pr(R_i \text{ is accessed}) = E[s_i(n)] = \mu_i(n)$. This implies that only the mean of the distribution of $\mathcal{S}(n)$ is needed in the computation of the total access probabilities of any record. All other characteristics of the distribution $f(\mathcal{S}(n))$ are not relevant for this purpose. With this clarification, two avenues for study are open. In the first one, we assume that although $f(\mathcal{S}(n))$ changes with time, the **means** of the distribution of $\mathcal{S}(n)$ are time invariant. In the subsequent model, we allow the means themselves to vary with time. These models are now presented in the increasing order of their generalities.

3.1 Markovian Multiplicative Model

Under this model of query generation, the probability distribution $\mathcal{S}(n+1)$ is derived from the distribution $\mathcal{S}(n)$ in a simple Markovian fashion as:

$$\mathcal{S}(n+1) = A^T \mathcal{S}(n),$$

where A is an $N \times N$ square matrix used for updating the distribution from one instant to another. It is easy to verify that the necessary and sufficient condition that the vector $\mathcal{S}(n+1)$ represent a probability distribution is that the square matrix A be stochastic.

With a more involved analysis it can be seen that this model of query generation is not one bit more powerful than the MSQG model, for indeed it is **exactly equivalent** to the MSQG model of query generation. To establish this correspondence, we simply let the matrix A represent the transition matrix of the MQSG. Under this mapping, the MSQG then possesses N states, where in state Q_i, element R_i is accessed with a probability of unity. With some insight we can see that this model generates the same distributions as those generated by the multiplicative model.

As a corollary it can be said that any MSQG system can be fully specified by at most N environments. This is because N states completely represent any multiplicative model. This fact is definitely nontrivial.

We now study the situation in which the distribution $\mathcal{S}(n)$ is a random vector. We first restrict the mean of $\mathcal{S}(n)$ to be the same for all values of the time variable n.

3.2 Time Invariant Means

Theorem 3 *Let $\mathcal{S}(n)$ be the query distribution used by the query generator at the time instant n. If $\mathcal{S}(n)$ has a distribution $f(\mathcal{S}(n))$ with a time invariant mean $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_N]^T$, then the following results hold for the MTF heuristic:*

1. *The Resulting Markov chain representing the data restructuring operation is homogeneous.*

2. *The steady state behaviour can be obtained by merely solving the homogeneous chain using $\boldsymbol{\mu}$ as the hypothetical query distribution.*

Proof:

Consider the fundamental Markov Chain associated with the problem and the MTF heuristic. In this chain, the states of the chain are all the permutations of the list. Let $L(n)$ denote the state of the chain at any time instant n. We proceed to examine the transition probabilities of this chain.

Let $< i_1, i_2, \ldots, i_N >$ and $< j_1, j_2, \ldots, j_N >$ denote any two permutations of the integers $\{1, 2, \ldots N\}$. Hence any general list configuration can be specified by the order $< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >$. The list is accessed at the time instant n according to the distribution $\mathcal{S}(n) = [s_1(n), s_2(n), \ldots, s_N(n)]^T$. Consider the transition from the order $< R_{j_1}, R_{j_2}, \ldots, R_{j_N} >$, to the order $< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >$. In the MTF heuristic, this transition is possible only if R_{i_1} is accessed and one of the following conditions are true:

$$(i) \quad i_m = j_m \quad \forall m = 1, 2, \ldots, N, \quad \text{or,}$$

$$(ii) \quad \exists\, m \text{ such that } i_1 = j_m \text{ and } < i_2, i_3, \ldots, i_N > = < j_1, j_2, \ldots, j_{m-1}, j_{m+1}, \ldots j_N > .$$

If none of the above conditions are satisfied, the MTF rule cannot produce a transition from the list $< R_{j_1}, R_{j_2}, \ldots, R_{j_N} >$ to the list $< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >$. In the following derivation, we shall restrict ourselves to only those combinations of the indices $< i_1, i_2, \ldots, i_N >$ and $< j_1, j_2, \ldots, j_N >$ for which the desired transition is possible. This is because of the fact that in cases where the choice of the indices $< i_1, i_2, \ldots, i_N >$, and $< j_1, j_2, \ldots, j_N >$ is such that a transition between the list configuration $< R_{j_1}, R_{j_2}, \ldots, R_{j_N} >$ to the desired state $< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >$ is not possible under the MTF heuristic, then this total transition probability simply evaluates to zero. Indeed, when such a transition is possible using the MTF rule, we have,

$$\begin{aligned} & Pr(L(n+1) =< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >| \ L(n) =< R_{j_1}, R_{j_2}, \ldots, R_{j_N} >, \mathcal{S}(n)) \\ &= Pr(R_{i_1} \text{ is accessed } | \text{ query distribution is } \mathcal{S}(n)) \\ &= s_{i_1}(n). \end{aligned} \tag{33}$$

The total probability of the going from the state $< R_{j_1}, R_{j_2}, \ldots, R_{j_N} >$ to the desired state $< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >$ (in cases where such a transition is possible) can be written using the laws of total probability and (33) as:

$$\begin{aligned} & Pr(L(n+1) =< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >| \ L(n) =< R_{j_1}, R_{j_2}, \ldots, R_{j_N} >) \\ &= \sum_{\text{all values of } \mathcal{S}(n)} s_{i_1}(n) \cdot f(\mathcal{S}(n)), \end{aligned} \tag{34}$$

where $f(\mathcal{S}(n))$ is the distribution of $\mathcal{S}(n)$. But the RHS of (34) is exactly $E[s_{i_1}(n)] = \mu_i(n)$. Since the means in our case are time invariant $E[s_{i_1}(n)] = \mu_{i_1}$. Hence,

$$Pr(L(n+1) =< R_{i_1}, R_{i_2}, \ldots, R_{i_N} >| \ L(n) =< R_{j_1}, R_{j_2}, \ldots, R_{j_N} >) = \mu_{i_1}.$$

Hence *all* the transition probabilities in the Markov chain of the MTF rule become time invariant and thus although the problem **itself** exhibits nonstationarity, the underlying Markov chain is homogeneous. Moreover, the transition probabilities are identical to those of a homogeneous Markov chain obtained if $\boldsymbol{\mu}$ was the stationary distribution. This proves the theorem. □

Notice that the derivation did not make any assumptions about the parametric form or the variance of the random vector $\mathcal{S}(n)$. Just the restriction of the time invariance of the **mean** of the random vector $\mathcal{S}(n)$ was sufficient to ensure the homogeneity of the resulting chain. The reader must notice that although $\mathcal{S}(n)$ itself is a random vector, the total probability that any specific record R_i is accessed, is time invariant. Thus although the result was proved only for the case of the MTF heuristic, the result holds for any time invariant list organizing heuristic.

3.3 Time Varying Means

Given that time invariant means are relatively easy to analyze, it is natural to investigate the consequences of having the means themselves vary with time. Let us denote the mean distribution used by the query generator at the time instant n be $\boldsymbol{\mu}(n)$. If $\boldsymbol{\mu}(n)$ is allowed to be an arbitrary function of n, the analysis is clearly intractable. Some insight into the problem could be obtained by suitably restricting the variation of $\boldsymbol{\mu}(n)$ with n. We now present our final result based on a family of functions which restricts the variation of $\boldsymbol{\mu}(n)$. The restriction we impose is very weak : all that we require is that the means themselves form a convergent sequence. However the result obtained is very general and includes all the known results involving the MTF rule.

Theorem 4 *Let $\mathcal{S}(n)$ be the (random) probability vector characterizing the accesses at time instant n which obeys the distribution $f(\mathcal{S}(n))$. Further $\boldsymbol{\mu}(n) = E[\mathcal{S}(n)]$. If additionally the following condition holds:*

$$\lim_{n\to\infty} \boldsymbol{\mu}(n) = \boldsymbol{\mu}^* = [\mu_1^*, \mu_2^*, \ldots, \mu_N^*]^T$$

then, under the MTF heuristic, the asymptotic probability, ${}_iP_j^*$ that record R_i precedes record R_j is given by:

$${}_iP_j^* = \frac{\mu_i^*}{\mu_i^* + \mu_j^*}.$$

Proof:

As before, we denote the probability that the record R_i precedes the record R_j at the time instant n by ${}_iP_j(n)$. We proceed in a manner analogous to that adopted for the case of the analysis of the MSQG model. In other words, we first construct a difference equation and examine its properties.

Now, under the MTF heuristic, the distribution of the *total* probability can be written in a recursive form, as follows:

$${}_iP_j(n+1) \quad = \quad \begin{cases} 1 & \text{w.p. } s_i(n) \\ 0 & \text{w.p. } s_j(n) \\ {}_iP_j(n) & \text{w.p. } 1 - s_i(n) - s_j(n). \end{cases} \tag{35}$$

Based on this equation, we can write down the *conditional expected value* of ${}_iP_j(n+1)$ as:

$$E[{}_iP_j(n+1)|{}_iP_j(n), \mathcal{S}(n)] = s_i(n) + (1 - s_i(n) - s_j(n)){}_iP_j(n). \tag{36}$$

In order to get $E[{}_iP_j(n+1)|\mathcal{S}(n)]$, we need to average the quantity $E[{}_iP_j(n+1)|{}_iP_j(n), \mathcal{S}(n)]$ using the distribution of ${}_iP_j(n)$. In doing so, we must note that the terms on the RHS of (36) involving components of $\mathcal{S}(n)$ are effectively constants. Hence, taking expectations of both sides of (36) a second time, we get,

$$E[{}_iP_j(n+1)|\mathcal{S}(n)] = s_i(n) + (1 - s_i(n) - s_j(n))E[{}_iP_j(n)]. \tag{37}$$

In order to get rid of the conditioning on the probability distribution $\mathcal{S}(n)$, we need to take expectations again. But the argument is a little intricate and is as follows.

The expected value of the LHS of (37) is $E[E[{}_iP_j(n+1)|\mathcal{S}(n)]] = E[{}_iP_j(n+1)]$. However the terms on the RHS of (37) need careful examination. The term $E[{}_iP_j(n)]$ only depends on the time instant $n-1$ and the past history. Hence, in averaging w.r.t. the distribution of $\mathcal{S}(n)$, this quantity can effectively be treated as a constant. This necessitates the use of the expectation operator one more time, and on applying the expectation operator for third time, we get,

$$\begin{aligned} E[{}_iP_j(n+1)] &= E[s_i(n)] + (1 - E[s_i(n)] - E[s_j(n)])E[{}_iP_j(n)] \\ &= \mu_i(n) + (1 - \mu_i(n) - \mu_j(n))E[{}_iP_j(n)]. \end{aligned} \tag{38}$$

Now the term $E[{}_iP_j(n)]$ simply represents the total probability that record R_i precedes record R_j at the time instant n. With this understanding, (38) can be rewritten as:

$${}_iP_j(n+1) = \mu_i(n) + (1 - \mu_i(n) - \mu_j(n)){}_iP_j(n). \tag{39}$$

We now wish to find the asymptotic value of ${}_iP_j(n)$ as $n \to \infty$. To derive this limit, we make use of the result that the mean $\boldsymbol{\mu}(n)$ converges to $\boldsymbol{\mu}^*$ as $n \to \infty$. Now, by the definition of limit, we have,

$$\forall \epsilon > 0 \quad \exists n_0 < \infty \quad \text{such that} \quad \forall n > n_0 \quad ||\boldsymbol{\mu}(n) - \boldsymbol{\mu}^*|| < \epsilon. \tag{40}$$

From (40), we deduce that the following condition holds whenever $n > n_0$:

$$|\mu_k(n) - \mu_k^*| \ < \epsilon/\sqrt{N}, \quad k = 1, 2, \ldots, N.$$

Equivalently,

$$\mu_k^* - \epsilon/\sqrt{N} < \ \mu_k(n) \ < \mu_k^* + \epsilon/\sqrt{N} \quad \forall n > n_0, \text{ and } \quad k = 1, 2, \ldots, N.$$

As in the proof of Theorem 2, let ${}_i\bar{P}_j(n)$ and ${}_i\underline{P}_j(n)$ denote upper and lower bounds on ${}_iP_j(n)$, respectively. Therefore the values of ${}_iP_j(n)$ derived from (39) for all values of $n \geq n_0$ are bounded from above and below by the solutions of the difference equations:

$${}_i\bar{P}_j(n+1) = (\mu_i^* + \epsilon/\sqrt{N}) + (1 - \mu_i^* - \mu_j^* + 2\epsilon/\sqrt{N}) \cdot {}_i\bar{P}_j(n) \tag{41}$$

$${}_i\underline{P}_j(n+1) = (\mu_i^* - \epsilon/\sqrt{N}) + (1 - \mu_i^* - \mu_j^* - 2\epsilon/\sqrt{N}) \cdot {}_i\underline{P}_j(n). \tag{42}$$

Notice that (41) and (42) are constant coefficient difference equations, whose asymptotic behaviour is easy to analyze. Following the steps identical to those involved in the latter part of the proof of Theorem 2 (omitted here in the interest of brevity) we obtain the desired result that:

$$_iP_j^* = \frac{\mu_i^*}{\mu_i^* + \mu_j^*}.$$

Hence the theorem. $\square$

4 Experimental Results

To validate the theoretical results derived in this paper, various experiments were conducted. The details of the experimental results are given in [12]. Some typical results are reported here.

In the first set of experiments which we report, the intention was to verify the results for the case of the MSQG model. The number of states in the Markov Chain for the query generator $\mathcal{Q}$ was fixed to be 3, for all experiments. The matrix of transition probabilities used in our experiments is:

$$\Delta = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.7 & 0.2 & 0.1 \\ 0.1 & 0.3 & 0.6 \end{bmatrix}$$

First, the number of elements in the list N was selected. Three random probability vectors (of dimension $N \times 1$) $\{\mathcal{S}_i | i = 1, 2, 3\}$ were generated for each experiment. When the query generator was in the state Q_i, the probability vector $\mathcal{S}_i$ was used for generating queries. Based on these probabilities and the steady-state probabilities of the chain characterized by Δ, theoretical values for the asymptotic probabilities $_iP_j^*$ were obtained. These probabilities were used to calculate the expected retrieval cost, based on (11). This quantity has been reported in Table 1.

To verify the accuracy of this cost, 50 experiments were run, each consisting of 10,000 time units (where one user query was generated per time unit). The retrieval cost was counted only during the last 5000 time units of each of these experiments. The ensemble average of the 50 time averages so obtained have also been reported in Table 1. In Table 1, we have presented results for the cases in which the number of records in the list was 5, 10, 15, and 20. In all the cases, we observe that the experimentally observed cost is very close to the theoretical cost. Thus, for example, the percentage difference between the theoretical and the observed values is only 0.3047%, for the case when $N = 20$.

In the second set of experiments, our aim was to verify the results presented for the RQG model. In these experiments, queries are generated at time n, based on the vector $\mathcal{S}(n)$ with a distribution $f(\mathcal{S}(n))$, whose mean changed with time. For the sake of brevity, we only report the results for the case when the number of elements in the list was 4. Also, in order

No. of elements in list	Theoretical Avg. Retrieval Cost	Observed Avg. cost	Error (in %)
5	2.2849	2.2421	1.8718
10	2.5015	2.4847	0.6726
15	2.3370	2.2890	2.0539
20	2.2973	2.2903	0.3047

Table 1: This table summarizes the results of the experiments conducted with the MSQG model of query generation. The Markov chain for the query generator had 3 states. The probability vectors used in these states were randomly generated. The table reports the theoretical expected asymptotic cost, as well as the experimentally observed retrieval cost, in cases when the number of elements in the list is 5, 10, 15, and 20.

to satisfy the conditions of Theorem 4, $\boldsymbol{\mu}(n)$ was chosen in such a way that it converged to the limiting value $\boldsymbol{\mu}^*$. To achieve this effect, we opted to generate successive generations of the **mean vector** $\boldsymbol{\mu}(n)$ according to the rule:

$$\boldsymbol{\mu}(n+1) = F^T \cdot \boldsymbol{\mu}(n). \tag{43}$$

The initial value $\boldsymbol{\mu}(0)$ was $[0.4, 0.3, 0.2, 0.1]^T$ and the limiting value $\boldsymbol{\mu}^*$ was $[0.3290, 0.2093, 0.2048, 0.2567]^T$.

Once again, 50 experiments were conducted, each involving 10,000 time units as mentioned earlier. The mechanism used for query generation at each time unit was as follows:

1. The value of the mean vector $\boldsymbol{\mu}(n)$ at the time instant n was determined according to the (43).

2. Let $\boldsymbol{\mu}(n) = [\mu_1(n), \mu_2(n), \ldots, \mu_N(n)]^T$. We generate a random vector $\mathcal{S}'(n) = [s'_1(n), s'_2(n), \ldots, s'_N(n)]^T$, where $s'_i(n)$ a random variable uniformly distributed in the open interval $(0, \mu_i(n))$. To transform $\mathcal{S}'(n)$ to be a probability vector, we merely normalize the components of $\mathcal{S}'(n)$, so as to sum to unity. This normalized vector is assigned to $\mathcal{S}(n)$, and it easy to see that $\mathcal{S}(n)$ has the mean $\boldsymbol{\mu}(n)$.

3. Generate a query based on the distribution $\mathcal{S}(n)$.

As before, only the last 5000 time units of each experiment were used in computing the average retrieval cost. The ensemble average of these 50 time averages is the observed expected retrieval cost and has a value of 2.468560. The theoretical expected retrieval cost in this case is 2.462485. The agreement between these values is remarkable.

5 Conclusions

In this paper, we have studied the problem of reorganizing the elements of a linear list according to the MTF heuristic. Most of the current literature about the MTF rule (in fact all the data structuring heuristics) only considers the case in which the user's query distribution is stationary, or time invariant.

Two models of nonstationary query distribution have been presented in this paper. The first is called the MSQG model for query generation — and is based on modeling the query generator *itself* as a Markov chain. In the second model, termed the RQG, the probability distribution used by the query generators at each time instant is itself a *random vector.* The MTF rule has been analyzed under both these models. The results generalize the corresponding known results for stationary environments and coincide astonishingly well with the intuitive generalization. The theoretical results derived in this paper have been experimentally verified.

References

[1] G.H. Gonnet, J.I. Munro and H. Suwanda, "Exegesis of Self-organizing Linear Search", *SIAM J. Computing*, Vol.10, No.3 (Aug. 1981), pp. 613-637.

[2] K.N. Hendricks, "The Stationary Distribution of an Interesting Markov Chain", *J. Appl. Prob.*, 9, 1, (Mar. 1972), pp. 231-233.

[3] K.N. Hendricks, "An Extension of a Theorem Concerning an Interesting Markov Chain", *J. Appl. Prob.*, 10, 4, (Dec. 1973), pp. 886-890.

[4] J.H. Hester and D.S. Hirschberg, "Self-organizing Linear Search", *ACM Computing Surveys*, Sept. 1985, Vol.17, No.3, pp. 295-311.

[5] D.E. Knuth, *The Art Of Computer Programming*, Vol.3, Sorting and Searching, Addision-Wesley, 1973.

[6] K.S. Narendra and M.A.L. Thathachar, *Learning Automata: An Introduction*, Prentice Hall, 1989.

[7] B.J. Oommen, and E.R. Hansen, "List organizing Strategies Using Stochastic Move-to-Front and Stochastic Move-to-rear operations", *SIAM J. Computing*, Vol.16, No.6, August 1987, pp.705-716.

[8] B.J. Oommen, E.R. Hansen, and J.I. Munro, "Deterministic Move-to-Rear List Organizing Strategies with Optimal and Expedient Properties", To appear in *Theoretical Computer Science.* A preliminary version of this paper was published in *Proc. Twenty-Fifth Annual Allerton Conference on Communication, Control and Computing*, Sept.-Oct.1987, Urbana, Illinois, pp.54-63.

[9] B.J. Oommen, and D.T.H. Ng, "Optimal Constant Space Move-to-Rear List Organization", *Proc. of the 1989 International Symposium on Optimal Algorithms*, Varna, Bulgaria, May-June 1989.

[10] R.L. Rivest, "On Self-Organizing Sequential Search Heuristics", *Comm. ACM*, Feb. 1976, Vol.19, No.2, pp.63-67.

[11] M.L. Tsetlin, "On the Behaviour of Finite Automata in Random Media", *Automation and Control*, Vol.22, 1962, pp.1210-1219. Originally appeared in *Automatika Telemekhanika*, Vol.22, 1961, pp.1345-1354.

[12] R.S. Valiveti, Ph.D. thesis, Carleton University. In preparation.

School of Computer Science, Carleton University
Bibliography of Technical Reports

SCS-TR-127	**On the Packet Complexity of Distributed Selection** A. Negro, N. Santoro and J. Urrutia, November 1987.
SCS-TR-128	**Efficient Support for Object Mutation and Transparent Forwarding** D.A. Thomas, W.R. LaLonde and J. Duimovich, November 1987.
SCS-TR-129	**Eva: An Event Driven Framework for Building User Interfaces in Smalltalk** Jeff McAffer and Dave Thomas, November 1987.
SCS-TR-130 Out of print	**Application Frameworks: Experience with MacApp** John R. Pugh and Cefee Leung, December 1987. Available in an abridged version in the Proceedings of the Nineteenth ACM SIGSCE Technical Symposium, February 1988, Atlanta, Georgia.
SCS-TR-131	**An Efficient Window Based System Based on Constraints** Danny Epstein and Wilf R. LaLonde, March 1988.
SCS-TR-132	**Building a Backtracking Facility in Smalltalk Without Kernel Support** See Third International Conference on OOPSLA, San Diego, Calif., Sept. '88. Wilf R. LaLonde and Mark Van Gulik, March 1988.
SCS-TR-133	**NARM: The Design of a Neural Robot Arm Controller** Daryl H. Graf and Wilf R. LaLonde , April 1988.
SCS-TR-134	**Separating a Polyhedron by One Translation from a Set of Obstacles** Otto Nurmi and Jörg-R. Sack, December 1987.
SCS-TR-135	**An Optimal VLSI Dictionary Machine for Hypercube Architectures** Frank Dehne and Nicola Santoro, April 1988.
SCS-TR-136	**Optimal Visibility Algorithms for Binary Images on the Hypercube** Frank Dehne, Quoc T. Pham and Ivan Stojmenovic, April 1988.
SCS-TR-137	**An Efficient Computational Geometry Method for Detecting Dotted Lines in Noisy Images** F. Dehne and L. Ficocelli, May 1988.
SCS-TR-138	**On Generating Random Permutations with Arbitrary Distributions** B. J. Oommen and D.T.H. Ng, June 1988.
SCS-TR-139	**The Theory and Application of Uni-Dimensional Random Races With Probabilistic Handicaps** D.T.H. Ng, B.J. Oommen and E.R. Hansen, June 1988.
SCS-TR-140	**Computing the Configuration Space of a Robot on a Mesh-of-Processors** F. Dehne, A.-L. Hassenklover and J.-R. Sack, June 1988.
SCS-TR-141	**Graphically Defining Simulation Models of Concurrent Systems** H. Glenn Brauen and John Neilson, September 1988
SCS-TR-142	**An Algorithm for Distributed Mutual Exclusion on Arbitrary Networks** H. Glenn Brauen and John E. Neilson, September 1988
	SCS-TR-143 to 146 are unavailable.
SCS-TR-147	**On Transparently Modifying Users' Query Distributions** B.J. Oommen and D.T.H. Ng, November 1988
SCS-TR-148	**An O(N Log N) Algorithm for Computing a Link Center in a Simple Polygon** H.N. Djidjev, A. Lingas and J.-R. Sack, July 1988 Available in STACS 89, 6th Annual Symposium on Theoretical Aspects of Computer Science, Paderborn, FRG, February 16-18, 1989, Lecture Notes in Computer Science, Springer-Verlag No. 349

SCS-TR-149	**Smallscript: A User Programmable Framework Based on Smalltalk and Postscript** Kevin Haaland and Dave Thomas, November 1988
SCS-TR-150	**A General Design Methodology for Dictionary Machines** Frank Dehne and Nicola Santoro, February 1989
SCS-TR-151	**On Doubly Linked List ReOrganizing Heuristics** D.T.H. Ng and B. John Oommen, February 1989
SCS-TR-152	**Implementing Data Structures on a Hypercube Multiprocessor, and Applications in Parallel Computational Geometry** Frank Dehne and Andrew Rau-Chaplin, March 1989
SCS-TR-153	**The Use of Chi-Squared Statistics in Determining Dependence Trees** R.S. Valiveti and B.J. Oommen, March 1989
SCS-TR-154	**Ideal List Organization for Stationary Environments** B. John Oommen and David T.H. Ng, March 1989
SCS-TR-155	**Hot-Spot Contention in Binary Hypercube Networks** Sivarama P. Dandamudi and Derek L. Eager, April 89
SCS-TR-156	**Some Issues in Hierarchical Interconnection Network Design** Sivarama P. Dandamudi and Derek L. Eager, April 1989
SCS-TR-157	**Discretized Pursuit Linear Reward-Inaction Automata** B.J. Oommen and Joseph K. Lanctot, April 1989
SCS-TR-158	**Parallel Fractional Cascading on a Hypercube Multiprocessor** Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, May 1989
SCS-TR-159	**Epsilon-Optimal Stubborn Learning Mechanisms** J.P.R. Christensen and B.J. Oommen, June 1989
SCS-TR-160	**Disassembling Two-Dimensional Composite Parts Via Translations** Doron Nussbaum and Jörg-R. Sack, June 1989
SCR-TR-161 (revised)	**Recognizing Sources of Random Strings** R.S. Valiveti and B.J. Oommen, January 1990 Revised version of SCS-TR-161 "On the Data Analysis of Random Permutations and its Application to Source Recognition", published June 1989
SCS-TR-162	**An Adaptive Learning Solution to the Keyboard Optimization Problem** B.J. Oommen, R.S. Valiveti and J. Zgierski, October 1989
SCS-TR-163	**Finding a Central Link Segment of a Simple Polygon in O(N Log N) Time** L.G. Alexandrov, H.N. Djidjev, J.-R. Sack, October 1989
SCS-TR-164	**A Survey of Algorithms for Handling Permutation Groups** M.D. Atkinson, January 1990
SCS-TR-165	**Key Exchange Using Chebychev Polynomials** M.D. Atkinson and Vincenzo Acciaro, January 1990
SCS-TR-166	**Efficient Concurrency Control Protocols for B-tree Indexes** Ekow J. Otoo, January 1990
SCS-TR-167	**A Hierarchical Stochastic Automaton Solution to the Object Partitioning Problem** B.J. Oommen, January 1990
SCS-TR-168	**Adaptive List Organizing for Non-stationary Query Distributions. Part I: The Move-to-Front Rule** R.S. Valiveti and B.J. Oommen, January 1990