TWO RESULTS ON SELF-ORGANIZING DATA STRUCTURES

James R. Bitner

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

TR-189 January 1982

Abstract: We show that the transposition rule is optimal over all permutation rules for the class of distributions where all but one element have the same probability. We also develop an interesting partial ordering on the costs of "reasonable" permutation rules, assuming the probability of the "odd" element is large. The rule with the greatest cost for this case is the move to front rule.

1. Introduction

The cost of accessing a key in a data structure depends on the "position" of the key. For example, accessing the ith key in an unordered list requires i probes. If the elements in the data structure are not equally likely to be accessed, the expected access time can be reduced by storing high probability keys in the positions of the data structure which can be accessed more quickly. In a list, the minimal cost is obtained by storing the keys in order of decreasing probability. However, suppose the probabilities are not known a priori. The obvious method of achieving a low cost ordering in a list is to count the number of accesses to each key and keep the list in order of decreasing counts. Clearly, this method is optimal but requires a (possibly large) amount of additional storage for the counters. Schemes using small, bounded amounts of additional storage have been studied (Gonnet, Munro, and Suwanda [1] and Bitner [2]) but we will confine our discussion to heuristics which do not use additional storage.

We also restrict our attention to lists (reordering schemes for trees have been studied by Bitner [2] and Allen and Munro [3]). A reordering scheme for a list which does not use additional storage is called a permutation rule and is defined by a set of n permutations $t_1,\ldots,t_n$. The ith permutation is applied to the list when the element currently in position i is accessed. One popular permutation rule is the move-to-front rule: If the element in position i is accessed, it is moved to position 1, and the element in positions 1 through i-1 are moved back one position. A second rule is the transposition rule: If the element in position i is requested, it is exchanged with the element in position i-1, unless i=1 in which case nothing is done.

We are primarily interested in the *cost of a rule* (defined precisely later) which is the expected number of probes required to access a randomly selected key after many accesses have been made. Though the list, of course, does not approach any given ordering, the probability of each ordering approaches its "steady state" probability. This allows us to speak of the (asymptotic) cost of a rule.

Permutation rules have been extensively studied [1, 2, 4-10]. One kind of result is to show a given rule is *optimal* (i.e. has lowest cost over a given class of rules and a given class of probability distributions). Rivest [7] has shown the cost of the transposition rule is less than that of the move to front rule for all non-trivial distributions and conjectured that the transposition rule is optimal over *all* rules and distributions. Simulation results (Rivest [7] and Tenenbaum [8]) support this claim. A. C. Yao (reported in [10]) proved that there is a probability distribution for which the transposition rule has lowest cost of all permutation rules. Hence, if there is a single rule which is optimal for all rules and distributions, it must be the transposition rule. Tenenbaum and Nemes [11] define for $1 \le k \le n-1$ a rule POS(k) which behaves as follows: when the i*th* element in the list is requested it is moved to position k if $i > k$ and is moved up one position if $i \le k$. For $2 \le k \le n$ they define the rule SWITCH(k) which behaves as follows: when the i*th* element is requested, it is moved up one position if $i > k$ and moved to the front if $i \le k$. They show POS(k) has lower cost than POS(k+1) and SWITCH(k) has lower cost than SWITCH(k+1) for the class of probability distributions where all but one element have equal probability. Since POS(1) and SWITCH(2) are the transposition rule, this result shows it to be optimal over this simple class of rules and distributions.

Our first result is of the same nature. We show (in Section 3) that the transposition rule is optimal for <u>all</u> rules over a restricted class of distributions where all but one element have the same probability.

A second kind of result is to define a <u>spectrum</u> of rules and prove a partial (or total) ordering on their costs. Clearly, the results in Tenenbaum and Nemes [11] are of this nature. Another possibility is to define A_i to be the rule where the requested key is moved up i positions (or to the front for keys in position 1 through i). It is conjectured (Bitner [2, 10]) that the rules $A_1, \ldots, A_n$ form a spectrum of rules where A_i has lower cost than A_{i+1}, but converges to that cost more slowly. This conjecture is supported by simulation results (Rivest [7] and Tenenbaum [8]).

In Section 4 we prove an extension of this result for the distributions we are studying assuming the probability of the "odd" element is large. We define a large class of rules (called reasonable rules) and show that if the number of permutations t_i that move the requested element to position 1 using a reasonable rule R exceeds the number for another reasonable rule Q, then rule R has higher cost than rule Q. This establishes a partial ordering on the costs of reasonable rules. A corollary of this result is that for such distributions the move to front rule has the highest cost of all reasonable rules.

2. Basic Definitions and Lemmas

In this section we give some basic definitions and notation and prove some lemmas about the cost of a permutation rule.

Notation: Throughout this paper, n denotes the number of elements in the list. The permutations for a given rule will be $t_1,\ldots,t_n$. Let the n-1 normal elements have probability p and the one odd element have probability q. We assume $q>p$. (Similar results hold if $q<p$). Finally, let $r=q/p$.

To analyze the cost of a given rule, we define a corresponding Markov chain and calculate its steady state cost. The chain consists of n states with state i corresponding to the arrangement with the odd element in position i. The transitions are defined as follows:

1. If $t_r(r) = i$ add a transition with probability q from state r to state i. (i.e. the odd element was in position r, was requested and moved to position i).
2. If $t_r(i) = j$ (for $i \neq r$) add a transition with probability p from state i to state j. (i.e. a normal element in position r was requested and this caused the odd element to move from position i to position j.)

This defines the Markov chain and its corresponding transition diagram, which is a directed multigraph with labeled edges. Note that there may be several transitions from one given state to another. Though their probabilities must be summed when defining the Markov chain, each is represented by a separate edge in the transition diagram.

Remark: Note that since each t_i is a permutation, the corresponding edges in the transition diagram form one or more cycles. One edge of this set of n edges has probability q and the others probability p.

It will also be important to reconstruct the permutation rule from the transition diagram (or, at least, to show there is some rule corresponding to this diagram).

Lemma 2.1: A transition diagram corresponds to a permutation rule if

1. The diagram has n edges labeled q, with no two having the same initial vertex and
2. The edges can be decomposed into n sets of n edges such that

   (a) each set has exactly one edge labeled q and
   (b) no two edges have the same initial vertex and
   (c) no two edges have the same final vertex

Proof: We construct the permutation rule corresponding to this diagram. The set with the edge labeled q emanating from state k will construct permutation t_k. (By condition 1 there is exactly one such edge for each k.) We choose $t_k(i)$ as the final state number of the edge whose initial state is i. Note that by conditions 2b and 2c, t_k is a permutation. By condition 2a, the transition $t_k(k)$ will have probability q and the rest probability p. The algorithm for constructing the diagram can now be used to show this permutation rule corresponds to this diagram. []

Definition: The steady state probability of a permutation rule R being in state i is denoted by $S_i(R)$. When it is clear what rule is being discussed, we will simply write S_i. We assume that, initially, every state is equally probable.

For most Markov chains, the steady state probabilities are independent of the initial distribution, but for some (called reducible or non-ergodic) it does. Specifying the initial distribution makes the S_i's defined for all chains.

The cost of a state is the expected number of probes required to access a randomly chosen element. The cost of rule R (COST(R)) is $\sum_{i=1}^{n} S_i$ * cost of state i. The following lemma computes COST(R).

Lemma 2.2: $COST(R) = n^2(n+1)p/2 + (q-p) \sum_{i=1}^{n} i\, S_i$

Proof: Consider state i. Since the element in position j (if $j \neq i$) is accessed with probability p and requires j probes and the element in position i is accessed with probability q and requires i probes, the cost of state i is

$$\left(\sum_{j \neq i} p * j\right) + q * i$$

$$= n(n+1)p/2 + (q-p)i$$

Multiplying by S_i and summing over i gives the total cost. []

Clearly, $\sum_{i=1}^{n} i\, S_i$ determines the cost for a given q. We give an explicit name to this quantity (the first moment) since it is easier to work with than the cost.

Definition: $MOM(R) = \sum_{i=1}^{n} i\, S_i\, (R)$

Lemma 2.3: Given two rules Q and R, COST(Q) < COST(R) iff MOM(Q) < MOM(R)

Next we give two lemmas for comparing the costs of two rules.

Lemma 2.4: Given two rules, Q and R, if there is a k such that

$S_i(Q) \geq S_i(R)$ for $1 \leq i \leq k$ and

$S_i(Q) \leq S_i(R)$ for $k+1 \leq i \leq n$

then $COST(Q) \leq COST(R)$

Proof:

$$
\begin{aligned}
& COST(Q) - COST(R) \\
&= \sum_{i=1}^{n} i\ (S_i(Q) - S_i(R)) \\
&= \sum_{i=1}^{k} i\ (S_i(Q) - S_i(R)) + \sum_{i=k+1}^{n} i\ (S_i(Q) - S_i(R)) \\
&\leq k \sum_{i=1}^{k} (S_i(Q) - S_i(R)) + k \sum_{i=k+1}^{n} (S_i(Q) - S_i(R)) \\
&= k \left(\sum_{i=1}^{n} S_i(Q) - \sum_{i=1}^{n} S_i(R) \right) = 0
\end{aligned}
$$

[]

Lemma 2.4 can also be found in Tenenbaum and Nemes [11]. We use it to prove the following, more useful lemma.

Lemma 2.5: Given 2 rules, Q and R, if

$S_i(Q)/S_{i+1}(Q) \geq S_i(R)/S_{i+1}(R)$

for all i, then $COST(Q) \leq COST(R)$

Proof: Let k be the smallest integer such that $S_i(Q) \leq S_i(R)$. Such a k must exist. If not, $S_i(Q) > S_i(R)$ for all i and at least one of the sets of probabilities would not sum to one. Therefore we have:

$$S_i(Q) > S_i(R) \qquad \text{for all } i < k$$

and

$$S_i(Q) \leq S_i(R) \qquad \text{for all } i \geq k$$

The second inequality is due to the fact that once $S_i(Q) \leq S_i(R)$ for some i, $S_j(Q) \leq S_j(R)$ for all $j>i$. This is easily shown from the "ratio property" in the condition of the lemma. []

3. The Optimality of the Transposition Rule

In this section we prove the transposition rule is optimal over the set of <u>all</u> permutation rules for the class of probability distributions where all but one element have the same probability. We assume <u>no</u> restrictions on the permutation rules we study; all rules, no matter how bizarre or unreasonable, are considered.

We begin by deriving n-1 steady state equations for a given Markov chain. For any $i<n$ consider the set of states 1 through i. Clearly, at steady state, the probability of a transition out of this set must equal the probability of a transition into this set. We have a transition <u>into</u> the set for all pairs (j,k) with $j>i$ such that $t_k(j) \leq i$ (i.e. a transition occurs from state j to some state with number at most i.) If $k=j$ this transition has probability q, else probability p. Similarly, there is a transition <u>out of</u> the set for every pair (j,k) with $j \leq i$ and $t_k(j)>i$ with probability q only if $k=j$. Therefore, we define the following <u>multisets</u>:

$JIQ(i) = \{k \mid j=k,\ j>i \text{ and } t_k(j) \leq i\}$
$JIP(i) = \{k \mid j \neq k,\ j>i \text{ and } t_k(j) \leq i\}$
$JOQ(i) = \{k \mid j=k,\ j \leq i \text{ and } t_k(j) > i\}$
$JOP(i) = \{k \mid j \neq k,\ j \leq i \text{ and } t_k(j) > i\}$

The names stand for <u>J</u>ump <u>I</u>n (or <u>O</u>ut) to the set of states 1 through i with probability <u>P</u> (or <u>Q</u>). In JIP(i) and JOP(i) there is one occurrence of k for every pair (j,k) satisfying the conditions. This gives us the steady state equations. For $i=1,\ldots,n-1$,

$$q_j \sum_{j \text{ in } JIQ(i)} S_j + p \sum_{j \text{ in } JIP(i)} S_j =$$

$$q_j \sum_{j \text{ in } JOQ(i)} S_j + p \sum_{j \text{ in } JOP(i)} S_j \qquad (3.1)$$

Before proceeding we note

$$| JIQ(i) | + | JIP(i) | = | JOQ(i) | + | JOP(i) |. \qquad (3.2)$$

This is because the edges created by each t_k can be partitioned into one or more cycles and the number of times a cycle leaves a set must equal the number of times it enters.

For most Markov chains these equations, plus the fact that the probabilities sum to one, determine the steady state probabilities, and we are guaranteed that the probability the chain is in a given state approaches that state's steady state probability. However if there are two sets of states and neither can reach the other (see Figures 3.1 and 3.2) the steady state probability depends on the initial distribution, and some of the n-1 steady state equations are redundant. To calculate the steady state probabilities, we solve the system of equations consisting of the steady state equations for each such set (there will be k-1 equations if there are k states in the set) and an equation stating that the sum of the steady state probabilities for the states equals the probability of the chain reaching some state in the set for the given initial distribution. In our case, this probability equals k/n, the probability of starting in this component.

Though it will be relatively simple to handle the case where some of the steady state equations are redundant, it is difficult if one disappears altogether (i.e. becomes 0=0). This motivates the following definition.

Definition: A Markov chain is <u>separable</u> iff for some i, JIP(i) = JIQ(i) = Ø. (Note this holds iff JOP(i) = JOQ(i) = Ø by equation (3.2).) Such an i is called a <u>separation point</u>. Suppose $i_1,\ldots,i_k$ are the separation points of a chain and let $i_0=0$ and $i_{k+1}=n$. Then for any j, the set of states i_j+1 through i_{j+1} is called an <u>inseparable component</u>.

Remark: The steady state equations for each separable component form a separate and independent system of equations.

We now define a property which, intuitively, a good rule must have:

Definition: A rule R is <u>monotonic</u> iff $S_i(R) \geq S_{i+1}(R)$ for all i.

To prove our main result, we first rule out non-monotonic rules, then show inseparable monotonic rules have higher cost than the transposition rule, and finally deal with separable monotonic rules.

Theorem 3.1: If R is not a monotonic rule, there exists a monotonic rule Q with COST(Q) $\leq$ COST(R).

Proof: Let $i_1,i_2,\ldots,i_n$ be a permutation of $1,2,\ldots,n$ such that $S_{i_1} \geq S_{i_2} \geq \ldots \geq S_{i_n}$. Redraw the transition diagram for R by renumbering each state k as i_k. This new transition diagram satisfies the conditions of Lemma 2.1. Hence it corresponds to a permutation rule which, by our renumbering of the states, is monotonic. COST(Q) $\leq$ COST(R) because $\sum_{i=1}^{n} i\, S_i$ is minimized over all permutations of some set of fixed S_i's by permuting them into decreasing order. []

Theorem 3.2: If R is an inseparable monotonic rule then $S_i(R)/S_{i+1}(R) \leq r$

Proof: Since the rule is monotonic, we have

$$q \sum_{j \text{ in } JIQ(i)} S_j + p \sum_{j \text{ in } JIP(i)} S_j$$

$$\leq q \sum_{j \text{ in } JIQ(i)} S_{i+1} + p \sum_{j \text{ in } JIP(i)} S_{i+1}$$

$$\leq S_{i+1} (q \mid JIQ(i) \mid + p \mid JIP(i) \mid) \qquad (1)$$

and

$$q \sum_{j \text{ in } JOQ(i)} S_j + p \sum_{j \text{ in } JOP(i)} S_j$$

$$\geq q \sum_{j \text{ in } JOQ(i)} S_i + p \sum_{j \text{ in } JOP(i)} S_i$$

$$\geq S_i (q \mid JOQ(i) \mid + p \mid JOP(i) \mid) \qquad (2)$$

From (3.1), (1) and (2) we have

$$S_i (q \mid JIQ(i) \mid + p \mid JIP(i) \mid)$$
$$\leq S_{i+1} (q \mid JOQ(i) \mid + p \mid JOP(i) \mid)$$

Note that $|JOQ(i)| + |JOP(i)| \neq 0$ because the chain is inseparable.

Therefore

$$\frac{S_i}{S_{i+1}} = \frac{q|JIQ(i)|+p|JIP(i)|}{q|JOQ(i)|+p|JOP(i)|}$$

$$\leq \frac{q|JIQ(i)|+q|JIP(i)|}{p|JOQ(i)|+p|JOP(i)|} \qquad \text{since } q \geq p$$

$$= q/p \text{ by } (3.2)$$ []

Theorem 3.3: Let T be the transposition rule,

then $S_i(T)/S_{i+1}(T) = r$

Proof: For the transposition rule, JIP(i) = {i+1}, JOP={i}, and JIP(i) = JOQ(i) = Ø. The steady state equations reduce to (for i = 1,...,n) q S_{i+1} = p S_i. []

Clearly, the result of the two previous theorems is the following.

Theorem 3.4: Let T be the transposition rule and R be any inseparable, monotonic rule, then COST(T) $\leq$ COST(R).

Proof: Immediate from Theorem 3.2, Theorem 3.3 and Lemma 2.5 []

Now we prove the most difficult theorem.

Theorem 3.5: If R is a separable monotonic rule and T is the transposition rule then COST(T) $\leq$ COST(R).

Proof: Given a separable, monotonic rule R, we construct another monotonic rule, R_3, such that COST(R) $\geq$ COST(R_3), and R_3 has fewer separation points. This process can be repeated until an inseparable monotonic rule is created. At this point Theorem 3.4 can be applied to show COST(T) $\leq$ COST(R_3), proving the theorem.

Let <u>k</u> be the smallest integer such that JIP(k) = JIQ(k) = Ø, and let <u>m</u> be the second smallest such number (or n if no such number exists). Define a rule R_1 which uses the transposition rule if element 1,...,k,k+2,...,m is accessed, does nothing if element k+1 is accessed and behaves the same as R for the other positions. R_1 has the same inseparable components as R. The contribution to the total cost by the first and second components is less for R_1. (Since we can calculate the cost for each component separately, we can apply Theorem 3.4 to these components in isolation). Further the contribution of the other components will be the same. Hence COST(R_1) $\leq$ COST(R).

We now define another rule, R_2, to be the same as R_1 except that when element k+1 is accessed it is transposed with element k. Note that R_2 has one fewer separation point than R_1. We have the following relations from the steady state equations (3.1):

$$S_i\ (R_1) = r\ S_{i+1}\ (R_1) \text{ for } i \text{ in } \{1,\ldots,m\}-\{k\} \tag{1}$$
$$S_i\ (R_2) = r\ S_{i+1}\ (R_2) \text{ for } i \text{ in } \{1,\ldots,m\} \tag{2}$$

Since states 1 through k and k+1 through m are separable components for R_1, we have

$$\sum_{i=1}^{k} S_i\ (R_1) = k/n \tag{3}$$

and

$$\sum_{i=k+1}^{m} S_i\ (R_1) = (m-k)/n \tag{4}$$

Since states m+1 through n are separable and identical components for both rules,

$$S_i\ (R_1) = S_i\ (R_2) \text{ for all } i > m \tag{5}$$

From (1), (2) and (5) we have

$$S_i(R_1)/S_{i+1}(R_1) \leq S_i(R_2)/S_{i+1}(R_2) \text{ for all } i \text{ in } \{1,\ldots,n\}-\{k\} \tag{6}$$

and if we can prove

$$S_k(R_1)/S_{k+1}(R_1) \leq S_k(R_2)/S_{k+1}(R_2) \tag{7}$$

Then we can apply Lemma 2.5 to prove $COST(R_1) \geq COST(R_2)$.

First, we have

$$S_k(R_2)/S_{k+1}(R_2) = r \tag{8}$$

from (2). We use (1) and (3) to solve for $S_k(R_1)$.

$$\sum_{i=1}^{k} S_i(R_1) = \frac{k}{n} = \sum_{i=1}^{k} r^{k-i} * S_k(R_1) = S_k(R_1) * \frac{r^k-1}{r-1}$$

and hence

$$S_k(R_1) = \frac{k}{n} * \frac{r-1}{r^k-1} \tag{9}$$

Similarly, we use (1) and (4) to obtain

$$S_{k+1}(R_1) = r^{m-k-1} * \frac{m-k}{n} * \frac{r-1}{r^{m-k}-1} \tag{10}$$

Then (7) is equivalent to

$$\frac{k(r^{m-k}-1)}{(m-k)\ r^{m-k-1}\ (r^k-1)} \leq r$$

which is equivalent to

$$k(r^{m-k}-1) \leq (m-k)\ r^{m-k}\ (r^k-1)$$

which is equivalent to

$$mr^{m-k} - (m-k)\ r^m - k \leq 0 \tag{11}$$

Now consider any fixed m and k $(m>k)$. We show (11) holds for all $r\geq 1$. Define the function $g(r) = mr^{m-k} - (m-k)\ r^m - k$. Clearly $g(1) = 0$. If we show $g'(r) \leq 0$ for all $r>1$, g will be monotonically non-increasing from zero for $r>1$. Since

$$\begin{aligned} g'(r) &= m(m-k)\ r^{m-k-1} - m(m-k)\ r^{m-1} \\ &= m(m-k)\ [r^{m-k-1} - r^{m-1}] < 0 \end{aligned}$$

(7) is established, proving $COST(R_2) \leq COST(R_1)$.

Finally, we define R_3 to be the result of permuting the states of R_2 as described in Theorem 3.1. Since the first component of R_2 is monotonic as is the rest, the effect will be to take some states from the first component of R_2 and insert them (while maintaining their relative order) between the states of the rest of R_2. This cannot increase the number of inseparable components. Hence R_3 is monotonic, has fewer inseparable components than R, and $COST(R_3) \leq COST(R)$ as desired. []

We now prove the main result.

Theorem 3.6: The transposition rule is optimal over all permutation rules for all distributions having n-1 elements with equal probability.

Proof: Let T be the transposition rule. From Theorem 3.4 and Theorem 3.5 we have $COST(T) \leq COST(R)$ for any monotonic rule R. For any non-monotonic rule, $\bar{R}_1$ there is a monotonic rule R' such that $COST(R') \leq COST(R)$ by Theorem 3.1. From the above, $COST(R') \leq COST(T)$ and the theorem is proved. []

4. A Partial Ordering over Reasonable Rules

In this section, we define a class of rules called "reasonable rules" and develop an interesting partial ordering on the costs of these rules, assuming r is large. The partial ordering divides the rules into classes with class i containing all rules which have i permutations that move the requested element to the front of the list. We show that all rules in class i have lower cost than all rules in class i+1 for large r (i.e. the more conservative a rule is the better).

This result is intuitive; if r is large the odd element will spend nearly all its time in the first position. Occasionally (with probability $|JOP(1)|*p$) a normal element will temporarily displace the odd element, which will then (almost certainly) return to first position on the next access. Thus $|JOP(1)|$ will be the dominant factor in determining the cost.

Definition: For rule R, let ONES(R) be the number of i's such that $t_i(i)=1$ and TWOS(R) be the number of i's such that $t_i(i)=2$.

Definition: A rule is <u>reasonable</u> if it satisfies for all $k>1$

$$t_k(k) < k$$
$$t_k(i) = i+1 \text{ for } t_k(k) \leq i < k$$
$$t_k(i) = i \text{ otherwise}$$

and (for $k=1$)

$$t_1(i) = i \text{ for all } i$$

In other words, the requested element is moved up in the list (except if it is in the first position), and all elements it passes over are moved back one position. No other element is moved.

Intuitively, reasonable rules are reasonable because the only way we "remember" our estimate of the probability of each element is by its position in the list. When an element is accessed it should be moved up to indicate that our estimate of its probability has increased. The relative order of the other elements should be left unchanged, since our estimate of their probabilities relative to each other remains unchanged.

The steady state equations also simplify considerably if we restrict ourselves to reasonable rules since JOQ(i) = JIP(i) = ∅ for any reasonable rule. Hence by (3.2) |JIQ(i)| = |JOP(i)| and (3.1) simplifies to

$$S_i = r \sum_{j \text{ in } JIQ(i)} S_j / |JIQ(i)| \qquad (4.1)$$

It will be convenient to deal with unnormalized probabilities defined by $u_i(R) = S_i(R)/S_n(R)$. Clearly,

$$u_n(R) = 1$$

$$u_i(R) = r \sum_{j \text{ in } JIQ(i)} u_j(R)/|JIQ(i)|$$

We can view each $u_i(R)$ as a polynomial in r that can be easily calculated from the $u_j(R)$'s with j>i. It is simply r times the average of the probability of the states in JIQ(i). (Note all elements of JIQ(i) are greater than i.) We also have

$$MOM(R) = \sum_{i=1}^{n} i\ u_i(R) / \sum_{i=1}^{n} u_i(R)$$

We need only calculate the leading coefficient of these polynomials.

Theorem 4.1: If R is a reasonable rule,

$$u_i = r^{n-i} / \prod_{j=i}^{n-1} |JIQ(j)| + O(r^{n-i-1})$$

Proof: We use induction for i=n,n-1,...,1. The theorem is trivially true for i=n, so suppose the theorem holds for all k>i. By (4.1) we have

$$u_i = r \sum_{j \text{ in } JIQ(i)} u_j / |JIQ(i)|$$

By induction $u_k = O(r^{n-i-2})$ for k>i+1;
therefore since i+1 is in JIQ(i)

$$u_i = r * (u_{i+1} + O(r^{n-i-2}))/|JIQ(i)|$$

and substituting inductively for u_{i+1} proves the theorem. []

Given the leading coefficients we can easily calculate MOM(R).

Theorem 4.2: If R is a reasonable rule

$$MOM(R) = 1 + |JIQ(1)| * r^{-1} + O(r^{-2})$$

Proof: By Theorem 4.1, u_i is an $n-i^{th}$ degree polynomial, so let

$$u_1 = br^{n-1} + cr^{n-2} + O(r^{n-3})$$

$$u_2 = dr^{n-2} + O(r^{n-3})$$

and $u_i = O(r^{n-3})$ for i>2. Then

$$MOM(R) = \frac{br^{n-1} + cr^{n-2} + 2dr^{n-2} + 0(r^{n-3})}{br^{n-1} + cr^{n-2} + dr^{n-2} + 0(r^{n-3})}$$
$$= 1 + (d/b) * r^{-1} + 0(r^{-2})$$

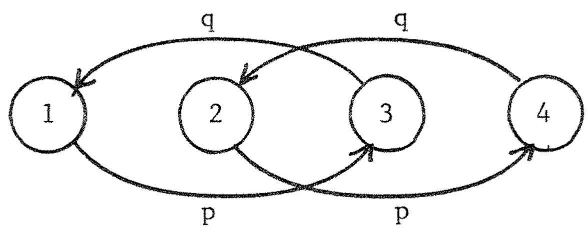
By Theorem 4.1, d/b = ¦JIQ(1)¦, which completes the proof. []

Theorem 4.3: For any n, there exists an r_0 such that for all $r>r_0$, and any two reasonable rules R and Q

if ONES(R) < ONES(Q) then COST(R) < COST(Q)

Proof: Since ONES(R) = ¦JIQ(1)¦ + 1 the result follows from Theorem 4.2 and Lemma 2.3. []

Remark: Theorem 4.3 can be extended to show that for any two reasonable rules R and Q, if ONES(R) = ONES(Q) but TWOS(R) < TWOS(Q) that COST(R) < COST(Q) for large r. The proof uses techniques similar to those used in Theorem 4.2, however the proof is much longer because we must consider the r^{n-1}, r^{n-2} and r^{n-3} terms. This requires extending Theorem 4.1 to calculate the second and third coefficients of the polynomial.



t_1 = (1 2 3 4)

t_2 = (1 2 3 4)

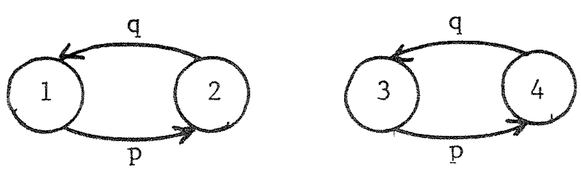
t_3 = (3 2 1 4)

t_4 = (1 4 3 2)

Figure 3.1

A permutation rule and its corresponding Markov chain.

NOTE: All transitions from a state to itself are not shown.



t_1 = (1 2 3 4)

t_2 = (2 1 3 4)

t_3 = (1 2 3 4)

t_4 = (1 2 4 3)

Figure 3.2

A separable Markov chain.

REFERENCES

1. G. H. Gonnett, J. I. Munro, H. Suwanda, Exegesis of Self Organizing Linear Search, SIAM J. Comput. 10 (1981) 613-637.
2. J. R. Bitner, Heuristics that Dynamically Organize Data Structures, SIAM J. Comput. 8 (1979) 82-110.
3. B. Allen and J. I. Munro, Self Organizing Search Trees, JACM 25 (1978) 526-535.
4. W. J. Hendricks, An Account of Self Organizing Systems, SIAM J. Comput. 5 (1976) 715-723.
5. D. E. Knuth, The Art of Computer Programming Vol. 3 Addison-Wesley, Don Mills Ont. 1973.
6. J. McCabe, On Serial Files with Relocatable Records, Operations Res. 12 (1965) 609-618.
7. R. Rivest, On Self-Organizing Sequential Search Heuristics, CACM 19 (1976) 63-67.
8. A. M. Tenenbaum, Simulations of Dynamic Sequential Search Algorithms, CACM 21 (1978) 790-791.
9. P. J. Burville and J. F. C. Kingman, On a Model for Storage and Search, J. Appl. Prob. 10 (1973) 697-701.
10. J. R. Bitner, Heuristics that Dynamically Alter Data Structures to Decrease their Access Time, Ph.D. Thesis, University of Illinois at Urbana-Champaign. TR UIUCSDCS-R-75-818. July 1976.
11. A. M. Tenenbaum and R. M. Nemes, Two Spectra of Self-Organizing Sequential Search Algorithms, SIAM J. Comput., to appear.