Programming Techniques
G. Manacher, S. Graham Editors

On Self-Organizing Sequential Search Heuristics

Ronald Rivest
Massachusetts Institute of Technology

This paper examines a class of heuristics for maintaining a sequential list in approximately optimal order with respect to the average time required to search for a specified element, assuming that each element is searched for with a fixed probability independent of previous searches performed. The "move to front" and "transposition" heuristics are shown to be optimal to within a constant factor, and the transposition rule is shown to be the more efficient of the two. Empirical evidence suggests that transposition is in fact optimal for any distribution of search probabilities.

Key Words and Phrases: searching, self-organizing, list-processing, heuristics

CR Categories: 3.74, 5.25

Heuristics for Maintaining a Sequential List

We consider heuristics for maintaining a sequential list in approximately optimal order, prove that the "transposition" heuristic is more efficient than the "move to front" heuristic analyzed by Knuth in [6, pp. 398–399], and present experimental evidence that the transposition heuristic is in fact optimal.

Suppose we have a set of n records $R_1, R_1, \ldots, R_n$ which we list in an arbitrary order π, so that R_i is in position $\pi(i)$ for $1 \leq i \leq n$. At each instant of time we are presented with a key K_i and asked to retrieve the associated record R_i. We do so by examining in turn each position of the list until R_i is found in position $\pi(i)$. This search costs $\pi(i)$ units of time to perform.

Let us assume that each key K_i is presented independently with probability p_i. The expected cost (average search length) for a permutation π is then

$$cost(\pi) = \sum_{1 \leq i \leq n} p_i \pi(i). \quad (1)$$

We assume without loss of generality throughout that $p_i \geq p_{i+1}$ for $1 \leq i \leq n$. Thus $cost(\pi)$ is minimized when π is the identity permutation, since the records are then in order of decreasing probability of being requested.

In practice, the relevant probabilities p_i are seldom known a priori, so that the optimal ordering can not be arranged in advance. A random initial arrangement can be expected to perform poorly, so we consider self-organizing schemes by which the initial ordering is gradually transformed on the basis of experience into a hopefully less costly arrangement.

A "counter" scheme immediately springs to mind, whereby we record the frequency f_i of requests for each record R_i, and maintain the records in order of decreasing frequency. By the strong law of large numbers, if $p_i > p_j$ we can expect $f_j > f_i$ to hold for at most a finite number of steps. Thus the counter scheme will stabilize on the optimal ordering.

The asymptotically optimal counter scheme requires extra memory space which, as Knuth remarks, could perhaps be better used by employing nonsequential search techniques. If using extra memory for counters is undesirable or not feasible, other self-organizing heuristics are available which tend to keep the list in near-optimal order.

The move to front heuristic has been studied by McCabe [7], Hendricks [3, 4], Burville and Kingman [1], and Knuth [6]. Schay and Dauer have studied a similar scheme [8]. Using this heuristic, whenever a record R_i is found in position $\pi(i)$, the list is rearranged by moving R_i to the front of the list and moving the records in positions $1, \ldots, \pi(i) - 1$ each down one position. The permutation π becomes $(1, 2 \ldots \pi(i))\ \pi$, the product of the cyclic permutation on the first i positions and π. Thus the records which are accessed frequently will tend to stay near the front of the list, while records infrequently accessed will drift towards the end. While no stable ordering is achieved, we can expect that near-optimal orderings will occur with high probability.

The average search time for this heuristic is easily calculated. Let $b(i, j)$ denote the asymptotic probability that R_i is before R_j in the list. This will be true at any

This work was performed at Institut de Recherche d'Informatique et d'Automatique, 78-Rocquencourt, France, and completed at MIT under NSF research grant no. GJ-43634X. Author's present address: MIT, Project MAC, NE43-804, 545 Technology Square, Cambridge, MA 02139.

time for which the most recent request for R_i has occurred since the most recent request for R_j. In this case, there exists a unique k such that the preceding k requests have been a request for R_i followed by $k-1$ requests for records other than R_i or R_j. Thus

$$b(i,j) = p_i \cdot \sum_{1 \le k < \infty} (1 - p_i - p_j)^{k-1} = p_i/(p_i + p_j). \tag{2}$$

The average search time is then

$$\sum_{1 \le j \le n} p_j \cdot \left(1 + \sum_{\substack{1 \le i \le n \\ i \ne j}} b(i,j)\right) = 1 + 2 \sum_{1 \le i < j \le n} p_i p_j/(p_i + p_j). \tag{3}$$

the result obtained in [6], [4], [1], and [7]. This counts only the search time; we should approximately double this if we wish to include the time to permute the elements after each search. A linked-list representation would allow these permutations to be made efficiently, but the space used for pointers might better be used for counters.

Let us consider how this scheme compares with the optimal ordering. Under our assumption that $p_i \ge p_{i+1}$ for $1 \le i \le n$, the minimal cost is

$$\sum_{1 \le j \le n} p_j \cdot j. \tag{4}$$

The ratio of (3) to (4) is then

$$\frac{1 + 2 \sum_{1 \le j \le n} p_j \cdot \sum_{1 \le i < j} p_i/(p_i + p_j)}{1 + \sum_{1 \le j \le n} p_j \cdot (j-1)} \tag{5}$$

$$\le (1 + 2x)/(1 + x) \quad \text{where } x = \sum_{1 \le j \le n} p_j \cdot (j-1) \tag{6}$$

$$\le 2(1 - 1/(n+1)) \quad \text{since } x \le \tfrac{1}{2}(n-1). \tag{7}$$

Thus the move to front scheme never does more than twice the work done with the optimal ordering. (An interesting open question is to determine the least constant c which is an upper bound to (5); the value 2 may not be the best possible.) This can be a considerable savings over a "random" initial ordering (average cost $= \frac{1}{2}(n+1)$) for typical probability distributions. For example, the optimal average cost for Zipf's distribution $p_i = i^{-1}H_n^{-1}$ (where H_n is the nth harmonic number) is just nH_n^{-1}, so that searching is $O(\ln n)$ times faster than with a random ordering. For Zipf's law Knuth shows, in addition, that with the move to front heuristic approximately 1.386 times the optimal cost is incurred asymptotically as $n \to \infty$.

We would like to know if we can do better. The generalized version of the above heuristic is to apply some permutation τ_j to the list after finding the desired record in position j. Is there a better choice for the τ's? We show that there is. We conjecture that the transposition heuristic is in fact optimal (that is, the best choice of τ's for *any* probability distribution). The transposition heuristic exchanges the desired record with the immediately preceding record; if the desired record heads the list nothing is done. Note that the transposition heuristic is simple to implement even if the list is stored in an ordinary one-dimensional array. We define a set of permutations τ to be *optimal* if for any probability distribution $\{p_i\}$ and any initial ordering $R_{i_1}, \ldots, R_{i_n}$ of the records, the expected cost of a search converges to a value $c(\tau)$ which is less than or equal to $c(\tau')$ for any other set of permutations τ'. The following theorem gives a partial characterization of an optimal set of τ's.

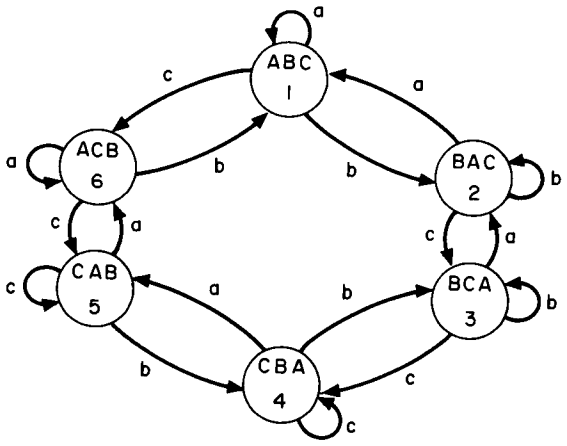
THEOREM 1. *An optimal set of permutations τ_j for $1 \le j \le n$ must have the property that each τ_j: (i) leaves positions $j+1$ to n of the list fixed, and (ii) moves record in position j to some position $j' < j$.*

PROOF. Consider the probability distribution $P_k : p_i = 1/k$ for $1 \le i \le k$ and $p_i = 0$ for $k < i \le n$, for some $k \le n$. Any set of permutations τ satisfying (i) and (ii) above will have an asymptotic cost of $\frac{1}{2}(k+1)$ since all of the records with zero probability will move to the end of the list and stay there. We prove (i) must hold by contradiction. Let τ be a set of permutations such that there are integers i, j such that $i < j$ and $\tau_i(j) < j$. Consider the probability distribution P_k with $k = j - 1$. Call an ordering $R_{i_1}, \ldots, R_{i_n}$ "good" if $p_{i_l} = 0$ for $l > k$. Clearly, under P_k any scheme which does not yield an asymptotic probability of zero for all non-good orderings will have a cost greater than $\frac{1}{2}(k+1)$ and will thus be non-optimal. Thus if τ were optimal, τ_i would be invoked with probability $1/k$, since position i is occupied by a record with probability $1/k$ in all good orderings. Yet if τ_i is applied to a good ordering, a non-good ordering necessarily results, since the record in position j, having zero probability, is moved in front of a record with nonzero probability. Thus non-good orderings would occur with frequency at least $1/k$. This contradiction proves that τ is non-optimal, proving that any optimal set τ must satisfy (i).

If τ satisfies (i) but not (ii), then there exists a j such that $\tau_j(j) \ge j$. By (i) we know that $\tau_i(j) = j$ for $i < j$. We can further conclude that $\tau_j(j) = j$ since $\tau_j(j) > j$ would cause $\tau_j(l) \ne l$ to occur for some $l > j$, thereby violating (i). Now consider an initial ordering $R_{i_1}, \ldots, R_{i_n}$ and corresponding probability distribution such that $p_{i_j} \ne 0$ and $p_{i_l} = 0$ for $l > j$. Clearly R_{i_j} will remain in position j. But if $p_{i_l} = 0$ for some $l < j$ in the initial ordering, then the records with zero probability will not end up at the end of the list, as any optimal scheme must do. Thus (ii) is also necessary in an optimal scheme. □

Note that we have not shown that an optimal set of permutations exist (that is, optimal for any probability distribution), but only what it must be like if it does. Theorem 1 also implies that under an optimal set of permutations every possible ordering of the records

Fig. 1.



which have nonzero probability will in fact occur with nonzero probability.

The analysis of the average cost of an arbitrary set of permutations τ is a straightforward if tedious task for any given set of probabilities p_i. We consider the finite Markov chain where each state is one of the $n!$ possible orderings of the set of records. The transition probabilities t_{ij} for the Markov chain are determined by the particular set of probabilities p and the permutations τ used; each t_{ij} will either be zero or else one of the p's. The stationary (asymptotic) probabilities of each state are the elements of the eigenvector of the matrix $T = \{t_{ij} \mid 1 \leq i \leq n! \text{ and } 1 \leq j \leq n!\}$ corresponding to the eigenvalue 1 (see [5]). The cost is then easily calculated from the formula

$$\sum_{\text{states } \pi} Prob(\pi) \cdot \sum_{1 \leq i \leq n} p_{\pi(i)} \cdot i. \tag{8}$$

As an example, consider a file of three records $R_1 = A$, $R_2 = B$, and $R_3 = C$ with respective probabilities of being requested of a, b, and c. Figure 1 shows the state diagram for the Markov system which results from using the transposition heuristic.

The transition matrix T for this system is then

$$T = \begin{bmatrix} a & a & - & - & - & b \\ b & b & a & - & - & - \\ - & c & b & b & - & - \\ - & - & c & c & b & - \\ - & - & - & a & c & c \\ c & - & - & - & a & a \end{bmatrix}. \tag{9}$$

The eigenvector of T corresponding to the eigenvalue 1 gives the stationary probabilities of this system. While these probabilities are difficult to compute symbolically as explicit functions of the variables a, b, and c, the eigenvector is easily calculated for any particular values. For example, if we take $a = .6$, $b = .3$, and $c = .1$ we obtain

$$\begin{aligned} Prob(A\ B\ C) &= 0.5, & Prob(B\ C\ A) &= 0.0417, \\ Prob(A\ C\ B) &= 0.167, & Prob(C\ A\ B) &= 0.0278, \\ Prob(B\ A\ C) &= 0.25, & Prob(C\ B\ A) &= 0.0139. \end{aligned} \tag{10}$$

The average search time for this system is 1.67, compared to 1.72 for the move to front heuristic with the same record request probabilities.

The reader may have noticed the rather remarkable relationship holding between the values in (10), given in the following theorem.

THEOREM 2. *Under the transposition heuristic the stationary probabilities obey:*

$$\frac{Prob(R_{i_1}R_{i_2} \cdots R_{i_j}R_{i_{j+1}} \cdots R_{i_n})}{Prob(R_{i_1}R_{i_2} \cdots R_{i_{j+1}}R_{i_j} \cdots R_{i_n})} = \frac{p_{i_j}}{p_{i_{j+1}}} \tag{11}$$

for $1 \leq j < n$ *if* $p_k \neq 0$ *for* $1 \leq k \leq n$.

PROOF. We note that if a set of state probabilities obeys (11), then it is a stationary distribution. For then we have

$$\begin{aligned} Prob(R_{i_1} \ldots R_{i_n}) &= p_{i_1} \cdot Prob(R_{i_1} \ldots R_{i_N}) \\ &\quad + \sum_{1 \leq j < n} p_{i_j} \cdot Prob(R_{i_1} \ldots R_{i_{j+1}}R_{i_j} \ldots R_{i_n}) \end{aligned} \tag{12}$$

which implies with (11) that

$$\begin{aligned} Prob(R_{i_1} \ldots R_{i_n}) &= [1/(1 - p_{i_1})] \cdot Prob(R_{i_1} \ldots R_{i_n}) \\ &\quad \cdot \sum_{1 \leq j < n} p_{i_j}(p_{i_{j+1}}/p_{i_j}) \\ &= Prob(R_{i_1} \ldots R_{i_n}). \end{aligned} \tag{13}$$

We also note that the set of inequalities (11) is consistent, since any sequence of transpositions leading from some ordering π of the records to another ordering π' will always yield the same ratio $Prob(\pi)/Prob(\pi')$. Since (11) is consistent and stationary, the stationary distribution must satisfy (11) since it is unique. □

Theorem 2 allows us to write a rather complicated expression for the average search time using the transposition heuristic. Let I_n denote the identity permutation on n letters $I_n(i) = i$ for $1 \leq i \leq n$. This is the optimal ordering under our assumption that $p(i) \geq p(i + 1)$ for $1 \leq i \leq n$. Let $\delta(i, \pi)$ denote the quantity $i - \pi(i)$ for any permutation π and $1 \leq i \leq n$; this is the number of places that R_i is displaced from its optimal position in π. Then the cost of the transposition heuristic is seen to be

$$Prob(I_n) \cdot \sum_{\pi} \left(\left(\prod_{1 \leq i \leq n} p_i^{\delta(i,\pi)} \right) \sum_{1 \leq j \leq n} p_j \cdot \pi(j) \right) \tag{14}$$

where the first sum ranges over all permutations π of the n records. We also have the following formula for $Prob(I_n)$:

$$Prob(I_n) = \left(\sum_{\pi} \prod_{1 \leq i \leq n} p_i^{\delta(i,\pi)} \right)^{-1} \tag{15}$$

We can now prove the following result, which was conjectured in [7].

THEOREM 3. *The transposition heuristic is always more efficient asymptotically than the move to front heuristic, except when $n = 2$ or all nonzero probabilities p_i are equal.*

PROOF. Consider the probability $b'(i, j)$ that R_i is before R_j under the transposition heuristic. We show that $b'(i, j) \geq b(i, j)$ for $1 \leq i < j \leq n$, with equality holding only for $n = 2$ or $p_i = p_j$, where $b(i, j)$ is the corresponding probability for the move to front heuristic. We then have

$$\frac{b'(i,j)}{b'(j,i)} = \frac{\sum_{k's} Prob(R_{k_1} \cdots R_{k_l} R_i R_{k_{l+1}} \cdots R_{k_m} R_j R_{k_{m+1}} \cdots R_{k_{n-2}})}{\sum_{k's} Prob(R_{k_1} \cdots R_{k_l} R_j R_{k_{l+1}} \cdots R_{k_m} R_i R_{k_{m+1}} \cdots R_{k_{n-2}})} \tag{16}$$

where the sums are taken over all permutations $k_1, \ldots, k_{n-2}$ of the integers $1, 2, \ldots, i-1, i+1, \ldots, j-1, j+1, \ldots, n$. But this gives immediately from Theorem 2:

$$b'(i, j)/b'(j, i) \geq \min_{1 \leq l < m \leq n} (p_i/p_j)^{m-l+1} = p_i/p_j \tag{17}$$

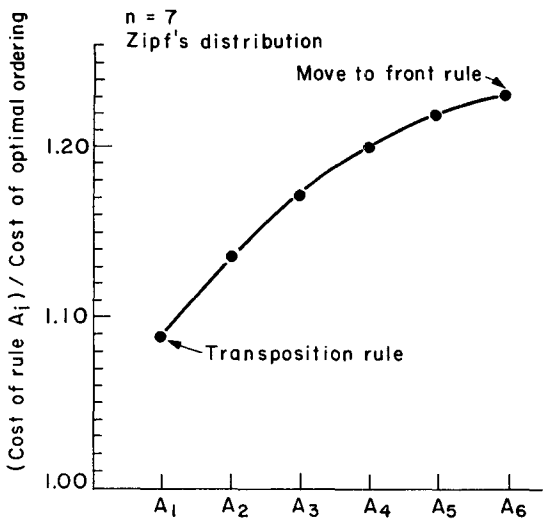
with equality holding only if $p_i = p_j$ or $n = 2$. When equality holds then $b'(i, j) = p_i/(p_i + p_j)$, the same as (2). Otherwise we have $b'(i, j) > b(i, j)$, which by the first part of (3) proves our theorem. □

We conjecture that the transposition heuristic is asymptotically more efficient than any other set of permutations for any probability distribution on the record requests. This conjecture is intuitively appealing since the transposition heuristic is the closest approximation obtainable to the optimal counter scheme, in a certain sense. Note that an element is advanced more than one position with the counter scheme only in the case where two or more of the preceding counters are equal. This is highly unlikely to happen often if all of the p_i's are distinct. In all other cases the counter scheme either performs a simple transposition or does nothing. The transposition heuristic is also the heuristic obeying (i) and (ii) of Theorem 1 which does the least damage to the previously established order, thus preserving the most information. We unfortunately lack a proof of this conjecture, however.

Andrew Yao [9] has shown that the transposition heuristic must be the optimal heuristic if there exists a single scheme which is optimal independent of the probability distribution. He shows that for the distribution $p_1 = 1 - (n-1)\epsilon$, $p_2 = p_3 = \ldots = p_n = \epsilon$ the transposition scheme is the only optimal scheme for ϵ sufficiently small. This lends strength to the conjecture.

Extensive simulation results for $3 \leq n \leq 12$ tend to confirm this conjecture. For example, Figure 2 plots the asymptotic cost of rule A_i over the optimal cost (4) for Zipf's distribution with $n = 7$ elements, where rule A_i is "move the desired record forward i places in the list, or to the front of the list if it was found in a position

Fig. 2.



$j < i$." Thus A_1 is the transposition rule and A_{n-1} is the move to front rule. The values plotted were obtained by simulating the search and rearrangement process for 5000 trials. A clear superiority of the transposition rule over the other rules is observed. While it is possible to consider other rules here, we have only plotted values for those rules which preserve the relative order of the unrequested elements, an intuitively reasonable feature.

For the case $n = 4$, all 72 sets of permutations τ satisfying (i) and (ii) of Theorem 1 were examined for several probability distributions. In every case the transposition heuristic was the most efficient. For example, with the geometric probability distribution $\bar{p} = (8/15, 4/15, 2/15, 1/15)$ the average search time varied between 1.94 and 2.24, with the transposition heuristic costing 1.94, the move to front rule costing 2.06, and the rule

$$\tau_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \quad \tau_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix},$$

$$\tau_3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 4 \end{pmatrix}, \quad \tau_4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$$

costing the most. The optimal cost for this system (given by (4)) is 1.73. These results were obtained by explicitly calculating the associated eigenvectors, instead of performing a simulation, since there are only 24 states to consider.

The main result of this paper concerns the asymptotic behavior of two self-organizing heuristics. A natural question to ask is, "what is the rate of convergence of these heuristics to their asymptotic efficiencies, given some initial ordering, or a probability distribution for initial orderings?", especially if one notes that the transposition heuristic generally takes more time to converge than the move to front if the records are initially in the reverse of their optimal ordering. This is perhaps an improbable way of beginning; a better model might be

one in which the list is initially empty and records which are searched for, but not found, are added to the end of the list. We can define a probability distribution for initial configurations as follows: let the algorithm begin with an empty table and insert new items as described above, until the table becomes full (and *not* using any self-organizing heuristics if a record is requested which is already in the table). The probability $Prob(\pi)$ that a given ordering π is arrived at when the last item is inserted into the table, will be our approximation to a reasonable initial probability distribution.

THEOREM 4. *The initial probability for a given ordering π is equal to the asymptotic probability of π occurring under the move to front heuristic.*

PROOF. The probability $b''(i, j)$ that R_i will be in front of R_j in the ordering is just the probability that a request for R_i occurred before a request for R_j in the list of requests. It is obvious that $b''(i,j) = p_i/(p_i + p_j)$ (as in eq. (2)), yielding the theorem (cf. eq. (3)). Note that our table-finding process is just the time dual of the move to front heuristic: R_i will be in front of R_j in the table-filling (move to front) process iff the first (last) request for R_i preceded (succeeded) the first (last) request for R_j. □

Under this model, the average efficiency of the initial states is equal to the average efficiency of the move to front heuristic, so that convergence for the move to front heuristic is instantaneous, but the transposition heuristic will be more efficient even from the beginning, since it yields an improvement from this state.

While the above result suggests that a possibly slower convergence to the asymptotic state by the transposition heuristic may not be a problem if the initial probabilities are arranged as indicated, it does not answer fully the question of the relative rate of convergence of these two strategies. This remains an open problem.

Conclusions

The transposition heuristic outperforms the move to front heuristic, and is quite likely the optimal heuristic for any probability distribution, given our assumption of the independence of requests. (If there is a high correlation between successive requests, one can show that the move to front heuristic is sometimes more efficient.)

A potential application for this result is the construction of list-processing systems such as LISP. Here one typically has a large number of names (atoms), each associated with a set of attribute-value pairs, such as "value," "printname," "function definition," and so on. A separate "property list" is usually maintained for each name listing these attributes and associated values. These property lists are repeatedly searched for the various variable values and function definitions during execution, and there is often not enough storage available to consider using a counter scheme. A dynamic optimization heuristic such as the transposition rule could be easily inserted into such a system, and would hopefully reduce the running time of an average program by a significant percentage at no extra cost in terms of storage. Many other similar applications are imaginable in the same kind of system where one has chosen a sequential list representation of the data for reasons of programming convenience, yet one wishes to reduce the search time at no extra cost in storage utilization. The transposition heuristic is an appealing choice in these situations due to its ease of implementation and demonstrated efficiency.

Another related application is the construction of paging algorithms for managing storage hierarchies. The list of pages concurrently in memory can be ordered using the transposition heuristic at each memory reference, and when a page fault occurs the last page on the list is replaced. The common "least recently used" scheme corresponds to the move to front heuristic described here. These schemes are studied by Franaszek and Wagner in [2].

It remains to be shown that the transposition heuristic is in fact optimal under our independence assumptions, or to find a counterexample to this conjecture. Another open problem is to examine the effects of relaxing the independence assumption upon the relative efficiency of various heuristics.

Acknowledgments. I would like to thank Prof. Donald E. Knuth for inspiring me to work on this problem and for his helpful suggestions on an earlier version of this paper. I have also learned from Knuth that Don Coppersmith (IBM Thomas J. Watson Research Center at Yorktown Heights) has some unpublished results of a similar nature. I would also like to thank Andrew Yao for his above-mentioned proof, as well as W.J. Hendricks and Colin McMaster for bringing additional references to my attention.

Received November 1974, revised March 1975

References

1. Burville, P.J., and Kingman, J.F.C., On a model for storage and search. *J. Appl. Prob. 10* (1973), 697–701.
2. Franasek, P.A. and Wagner, T.J. Some distribution-free aspects of paging algorithm performance. *J. ACM 211* (Jan. 1974), 31–39.
3. Hendricks, W.J. The stationary distribution of an interesting Markov chain. *J. Appl. Prob. 9* (1972), 231–233.
4. Henricks, W.J. An extension of a theorem concerning an Interesting Markov chain. *J. Appl. Prob. 10* (1973), 886–890.
5. Kemeny, J.G., and Snell, J.L. *Finite Markov Chains.* Van Nostrand, Princeton, N.J., 1960.
6. Knuth, D.E. *The Art of Computer Programming, Vol. 1, Sorting and Searching.* Addison-Wesley, Reading, Mass., 1973.
7. McCabe, J. On serial files with relocatable records. *Operations Res. 12* (July 1965), 609–618.
8. Schay, G. Jr., and Dauer, F.W. A probabilistic model of a self-organizing file system. *SIAM J. Appl. Math. 15* (1957), 874–888.
9. Yao, A. Personal communication.