ON SELF-ORGANIZING SEQUENTIAL SEARCH HEURISTICS

Ronald L. RIVEST.*

Abstract :

We examine 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 we search for each element 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.

Keywords : searching, self-organizing, list-processing, heuristics.

We consider heuristics for maintaining a sequential list in approximately optimal order, prove that the "transposition" heuristic is more efficient that the "move to front" heuristic analyzed by Knuth [2,pp. 398-399], and present experimental evidence that the transposition heuristic is in fact optimal.

Suppose we have a set of n records $R_1, R_2, \ldots, R_n$ which we list in an arbitrary order π, so that R_i is in position $\pi(i)$ for $1 \le i \le n$. 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

$$\mathrm{cost}(\pi) = \sum_{1 \le i \le n} p_i \pi(i) \qquad (1)$$

We assume without loss of generality throughout that $p_i \ge p_{i+1}$ for $1 \le i \le n$. Thus $\mathrm{cost}(\pi)$ 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 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.

*IRIA, 78150 Rocquencourt, France.

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 is apparently of unknown origin, although Schay and Dauer studied a similar scheme [3]. 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 $\pi(i)$ 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 near-optimal orderings to 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 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

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

The average search time is then

$$\sum_{1 \le j \le n} p_j \cdot \Big(1 + \sum_{\substack{1 \le i \le n \\ i \ne j}} b(i,j)\Big) = 1 + 2 \sum_{1 \le i < j \le n} \frac{p_i p_j}{p_i + p_j} \qquad (3)$$

the result obtained by Knuth [2, equation 6.1(17)].

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 $p_i \geq p_{i+1}$ for $1 \leq i \leq n$, the minimal cost is

$$\sum_{1 \leq j \leq n} p_j \cdot j \qquad (4)$$

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

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

$$\leq \frac{1 + 2x}{1 + x} \quad \text{where } x = \sum_{1 \leq j \leq n} p_j \cdot (j-1) \qquad (6)$$

$$\leq 2\left(1 - \frac{1}{n+1}\right) \text{ since } x < \frac{n-1}{2} . \qquad (7)$$

Thus the "move to front" scheme never does more than twice the work done with the optimal ordering. This can be a considerable savings over a "random" initial ordering (average cost = $(n+1)/2$) 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 n-th harmonic number) is just $n H_n^{-1}$, so that searching is $O(\ln(n))$ times faster that 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. The following theorem gives a partial characterization of an optimal set of τ's.

Theorem 1. *An optimal set of permutations τ_j for $1 \leq j \leq n$ must have the property that each τ_j :*

(i) leaves positions $j+1$ to n of the list fixed, and

(ii) moves the record in position j to some position $j' < j$.

Proof : Consider the probability distribution $p_i = 1/k$ for $1 \leq i \leq k$ and $p_i = 0$ for $k < i \leq n$, for some $k < n$. Any set of permutations τ satisfying (i) and (ii) above will have an asymptotic cost of $(k+1)/2$ since all of the records with zero probability will move to the end of the list and stay there. Any set of permutations τ which violates (i) will occasionally move a record with zero probability in front of one with non-zero probability, and thus have greater average cost. Any set of permutations τ which satisfies (i) but not (ii) will not be able to move any records out of positions j such that $\tau_j(j) = j$, so that the optimal ordering for our particular probability distribution can not be reached, again incurring a higher cost. □

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. The above theorem also implies that under an optimal set of permutations every possible ordering of the records which have non-zero probability will in fact occur with non-zero 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 [1]). The cost is then easily calculated from the formula :

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

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.

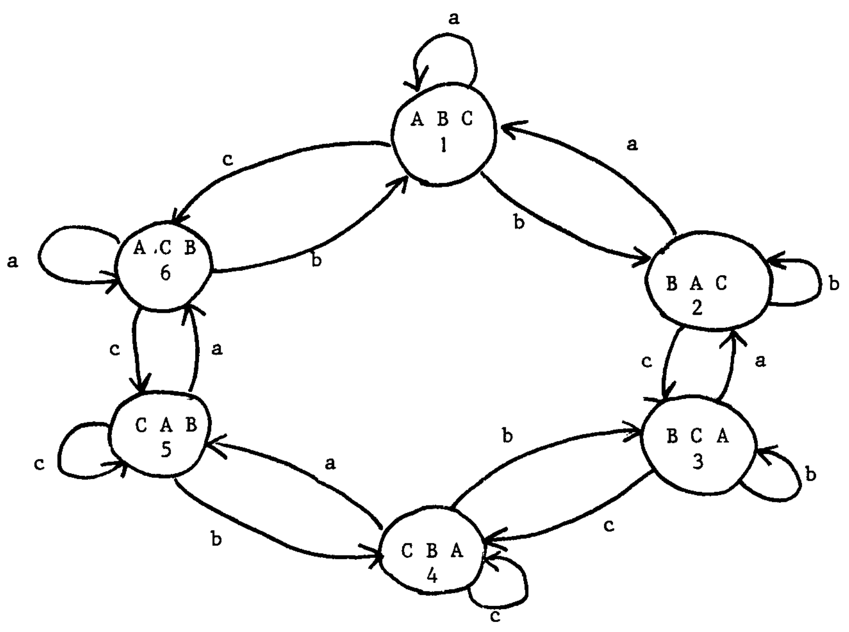

Figure 1.

The transition matrix T for this system is then :

$$T=\begin{bmatrix} a & a & - & - & - & b \\ b & b & a & - & - & - \\ - & c & b & b & - & - \\ - & - & c & c & b & - \\ - & - & - & a & c & c \\ c & - & - & - & a & a \end{bmatrix} \qquad (9)$$

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, for any particular values the eigenvector is easily calculated. For example, if we take a =.6, b =.3, and c =.1 we obtain

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

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 :*

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

for $1\le j<n$ if $p_k \ne 0$ for $1\le k\le n$.

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

$$\text{Prob}(R_{i_1}\dots R_{i_n}) = p_{i_1}.(\text{Prob}(R_{i_1}\dots R_{i_n}) + \text{Prob}(R_{i_2}R_{i_1}\dots R_{i_n})) + \sum_{2\le j<n} p_{i_j}.\text{Prob}(R_{i_1}\dots R_{i_{j+1}}R_{i_j}\dots R_{i_n}) \qquad (12)$$

which implies with (11) that

$$\text{Prob}(R_{i_1}\dots R_{i_n}) = \frac{1}{1-p_{i_1}-p_{i_2}}.\text{Prob}(R_{i_1}\dots R_{i_n}).\sum_{2\le j<n} p_{i_j}.\frac{p_{i_{j+1}}}{p_{i_j}} = \text{Prob}(R_{i_1}\ \dots\ R_{i_n}). \qquad (13)$$

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 $\text{Prob}(\pi) / \text{Prob}(\pi')$. Since (11) is consistent and stationary, the stationary distribution must satisfy (11) by the uniqueness of the stationary distribution.□

The preceding theorem 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 $I(i) = i$ for $1\le i\le n$. This is the optimal ordering under our assumption that $p(i) \ge p(i+1)$ for $1\le i\le n$. Let $\delta(i,\pi)$ denote the quantity $i - \pi(i)$ for any permutation π and $1\le i\le n$; 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

$$\mathrm{Prob}(I_n) \cdot \sum_{\pi} \prod_{1 \le i \le n} p_i^{\delta(i,\pi)} \sum_{1 \le j \le n} p_j \cdot \pi(j) \qquad (14)$$

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

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

We can now prove the following theorem.

Theorem 3 : *The "transposition" heuristic is always more efficient asymptotically than the "move to front" heuristic, except for the case $n = 2$ or the case where all non-zero 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 $b'(i,j) \ge b(i,j)$ for $1 \le i < j \le n$, 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

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

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

$$\frac{b'(i,j)}{b'(j,i)} \ge \min_{1 \le l < m \le n} \left(\frac{p_i}{p_j} \right)^{m-l+1} = \frac{p_i}{p_j} \qquad (17)$$

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. □

The preceding theorem is the main theoretical result presented here. We conjecture that the transposition heuristic is asymptotically more efficient than any other set of permutations for any probability distribution for 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 that 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 are unfortunately lacking a proof of this conjecture, however.

Externsive simulation results for $3 \le n \le 12$ 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 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 5,000 trials. A clear superiority of the transposition rule over the other rule is observed. While there are other rules possible to consider 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 $\tau_1 = \binom{1\ 2\ 3\ 4}{1\ 2\ 3\ 4}$ $\tau_2 = \binom{1\ 2\ 3\ 4}{2\ 1\ 3\ 4}$, $\tau_3 = \binom{1\ 2\ 3\ 4}{3\ 2\ 1\ 4}$, $\tau_4 = \binom{1\ 2\ 3\ 4}{3\ 2\ 4\ 1}$ costing the most. The optimal cost for this system (given by (4)) is 1.73. These results were obtained by explicity calculating the associated eigenvectors, instead of performing a simulation, since there are only 24 states to consider.

Conclusions.

The transposition heuristic outperforms the "move to front" heuristic, and is perhaps the best candidate for an optimal heuristic for any probability distribution, given our assumption of the independence of requests. (If there is high correlation between sucessive 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.

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

Acknowledgement.

I would like to thank Prof. Donald E. Knuth for his comments on an earlier version of this paper, and for inspiring me to work on this problem. I have also learned from Knuth that Don Coppersmith (IBM Research at Yorktown Heights) has some unpublished independent results of a similar nature.

References.

[1] Kemeny, John G. and J. Laurie Snell. Finite Markov Chains. D. Van Nostrand Co., Inc. Princeton, New Jersey (1960).

[2] Knuth, Donald E. The Art of Computer Programming (volume 3 : Sorting and Searching). Addison-Wesley Publishing Co. Reading, Mass. (1973).

[3] Schay, G., Jr., and F. W. Dauer. A Probabilistic Model of a Self-Organizing File System. SIAM J. Appl. Math. 15 (1967), 874-888.

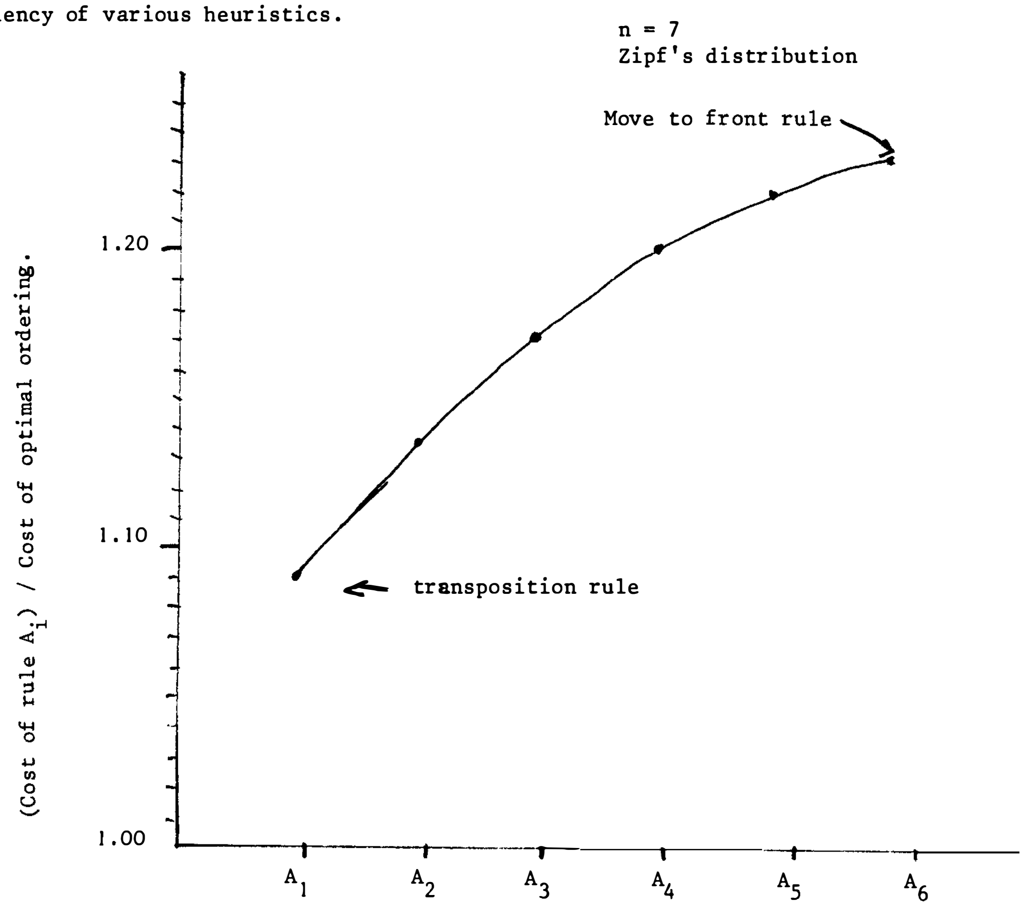

Figure 2. Relative efficiency of rules $A_1, \ldots, A_6$.