AN OPTIMAL LIST ORGANIZING STRATEGY WHICH USES STOCHASTIC MOVE-TO-FRONT OPERATIONS[†]

B. J. Oommen*

SCS-TR-53

June 1984

[†]Supported by the Natural Sciences and Engineering Research Council of Canada.

*School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6

AN OPTIMAL LIST ORGANIZING STRATEGY WHICH USES STOCHASTIC MOVE-TO-FRONT OPERATIONS*

B. John Oommen+

ABSTRACT

Consider a list of elements $\{R_1,\ldots,R_N\}$ in which the element R_i is accessed with an (unknown) probability s_i. If the cost of accessing R_i is proportional to i (as in sequential search) then it is advantageous if each access is accompanied by a simple reordering operation. This operation is chosen so that ultimately the list will be sorted in the descending order of the access probabilities.

We present a dynamic list organizing scheme which permits the element R_j to be moved to the front of the list on being accessed. However, as opposed to the schemes discussed in the literature, the move operation is performed with probability p_j. The latter quantity adaptively evolves in such a way that in the limit, no more move operations are performed. When this occurs we say that the scheme has converged. Although the scheme could converge to any one of the N! possible configurations, we shall show that by suitably updating the probabilities $\{p_j\}$ the probability of converging to the right arrangement can be made as close to unity as desired.

---

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

+ School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6.

Key Words: Dynamic List Ordering, Move to Front Rule, Adaptive Learning, Self-Organizing Lists, Stochastic List Operations.

I. Introduction

Suppose we are given a set of elements $\{R_1,\ldots,R_N\}$. At every time instant one of these elements is accessed. Further, the element R_j is accessed with an unknown access probability s_j. We assume that the accesses are made independently. Whenever an element R_j is accessed, a sequential search is performed on the list. To minimize the cost of accessing, it is desirable that the records are ordered in the descending order of their access probabilities. We shall refer to a file ordered in this way as a completely organized file.

McCabe [11, pp.398-399; 12] was the first to propose a solution to this problem. His solution rendered the list dynamically self-organizing and it involved moving an element to the front of the list every time it was accessed. Using this rule, the Move-to-Front (MTF) rule, the limiting average value of the number of probes done per access has the value C_{MTF}, where,

$$C_{MTF} = 0.5 + \sum_{i,j} (s_i\, s_j)/(s_i + s_j)$$

Many other researchers [3, 6-9,14] have also extensively studied the MTF rule and various properties of its limiting convergence characteristics are available in the literature.

McCabe [11, pp.398-399, 12] also introduced a scheme which is called the transposition rule. This rule requires that an accessed element is interchanged with its preceding element in the list, unless, of course, it is at the front of the list. Much literature is available on the transposition rule [1,2,4,11,12,13] but particularly important is the work of Rivest [13] and Tenenbaum et al [1,15] who extensively studied this rule

and suggested its generalizations - the Move-k-Ahead rule and the POS(k) rule. In the former, the accessed element is moved k-positions forward to the front of the list unless it is found in the first k positions - in which case it is moved to the front. The POS(k) rule moves the accessed element to position k of the list if it is in positions k+1 through n. It transposes it with the preceding element if it is in positions 2 through k. If it is the first element it is left unchanged.

Rivest [13] showed that the limiting behaviour of the transposition rule (quantified in terms of the average number of probes) was never worse than that of the MTF rule. He conjectured that the transposition rule has lower expected cost than any other reorganization scheme. He also conjectured that the move-k-ahead rule was superior to the move-k+1-ahead rule; but as yet this is unproven. Tenenbaum and Nemes [15] proved various results for the POS(k) rule primarily involving a distribution in which $s_2 = s_3 = \ldots = s_N = (1-s_1)/(N-1)$. Their results seem to strengthen Rivest's conjecture.

All of the schemes discussed in the literature are represented by Markov chains which are ergodic. By virtue of this fact, the list can be in any one of its N! configurations - even in the limit. Thus, for example, a list which was completely organized can be thoroughly disorganized by the MTF rule by a <u>single</u> request for the element which is accessed most infrequently. Observe that after this unfortunate occurrence, it will take a long time for the list to be organized again - i.e., for this element to dribble its way to the tail of the list.

In this paper we propose a learning algorithm in which the elements of the list adaptively learn to find their place. The algorithm is essentially a move-to-the-front algorithm with the exception that an element R_j is moved to the front of the list with a probability p_j. This probability is systematically decreased every time the element is accessed. Ultimately on being accessed, each element tends to stay in the place where it is (as opposed to moving to the front of the list). In other words, the markovian representation of the procedure is absorbing, as opposed to ergodic. The organization of the list gets "absorbed" into one of the N! orderings. However, we shall show that the probability of being absorbed into the optimal arrangement can be made as close to unity as desired.

Apart from the algorithm presented by us being much more accurate than all the algorithms reported in the literature, it is computationally more efficient. This is because the access of an element requires the update of <u>exactly</u> one probability (namely the probability p_j associated with the accessed element R_j). Further, unlike the contemporary algorithms, since the list operations are essentially stochastic, a list operation is <u>not</u> necessarily performed on every access. Finally, since the markovian representation of the scheme is absorbing, the number of list operations performed asymptotically decreases to zero.

Simulation results for various distributions of $\{s_j\}$ indeed demonstrate the optimal properties of the scheme proposed.

As in the literature concerning the theory of adaptive learning, we shall use the terms algorithm, rule and scheme interchangeably.

II. Probabilistic Move-to-Front Operations

The concept of performing probabilistic move operations on an accessed element is not entirely new. Kan and Ross [9] suggested a probabilistic transposition scheme and showed that no advantage was obtained by rendering the scheme probabilistic. Their scheme, however, required that the probability of performing the operation, be time invariant. As opposed to this, we shall define move operations which are essentially probabilistic, but the probabilities associated with the move operations are dynamically varied.

Let $p_i(n)$ be the probability (at time 'n') of the element R_i being moved to the front of the list on being accessed. Observe that this implies that the element, on being accessed, stays where it is with probability $(1-p_i(n))$. For an initial condition we define,

$$p_i(0) = 1, \qquad i = 1,\ldots,N. \tag{1}$$

The probability $p_i(n)$ is updated every time the record R_i is accessed. The updating scheme is given by (2) below, for $0<a<1$.

$$\begin{aligned} p_i(n+1) &= ap_i(n) && \text{if } R_i \text{ is accessed} \\ &= p_i(n) && \text{if any other } R_j \neq R_i \text{ is accessed.} \end{aligned} \tag{2}$$

The quantity 'a' is defined as the updating constant. We shall later discuss how this constant is chosen. Observe that the updating scheme is reminiscent of the Linear Reward-Inaction scheme studied extensively in the area of adaptive learning.

We now derive the properties of $p_i(n)$. To simplify notation, unless explicitly stated, p_i will refer to the quantity $p_i(n)$.

Theorem I

$E[p_i(n)]$ decreases monotonically with time. Further, for all $j \neq k$,

$s_j > s_k$ if and only if $E[p_j(n)] < E[p_k(n)]$.

Proof: Consider the random variable $p_i(n+1)$. By virtue of (2), the latter has the following distribution:

$$\begin{aligned} p_i(n+1) &= ap_i, \quad \text{w. prob.} \quad s_i \\ &= p_i \quad \text{w. prob.} \quad (1-s_i) \end{aligned} \tag{3}$$

$$\begin{aligned} \text{Thus, } E[p_i(n+1)|p_i] &= ap_is_i+p_i(1-s_i) \\ &= p_i(1-(1-a)s_i) \\ &= e_ip_i, \quad \text{where } e_i=1-(1-a)s_i. \end{aligned} \tag{4}$$

Note that since $0<a<1$, for all i, e_i obeys $0<e_i<1$. Taking expectations again, we obtain,

$$E[p_i(n+1)] = e_iE[p_i(n)]. \tag{5}$$

The difference equation (5) subject to the initial condition (1), yields the solution:

$$E[p_i(n)] = e_i^n.$$

Clearly, $E[p_i(n)]$ is monotonically decreasing with n. Further,

$$s_j > s_k \iff e_j < e_k \iff e_j^n < e_k^n ,$$

and the theorem is proved. ***

Corollary 1. For all i,

$$\lim_{n\to\infty} p_i(n) = 0 \qquad \text{w. prob. 1.}$$

The result follows since the Markov process $\{p_i(n)\}$ has only one absorbing barrier, namely the probability 0 [10].

Theorem II

For $n>0$, Var $[p_i(n)]$ decreases monotonically with n and its limiting value is zero.

Proof: From the distribution of $p_i(n)$ given by (3), we obtain,

$$p_i^2(n+1) = a^2 p_i^2 \qquad \text{w. prob. } s_i$$
$$= p_i^2 \qquad \text{w. prob. } (1-s_i).$$

Thus, $E[p_i^2(n+1)|p_i] = [1-(1-a^2)s_i]p_i^2$. (6)

Taking expectations again we obtain the difference equation,

$$E[p_i^2(n+1)] = [1-(1-a^2)s_i]\, E[p_i^2(n)].$$

Since $E[p_i^2(0)] = 1$, we have,

$$E[p_i^2(n)] = [1-(1-a^2)s_i]^n \ . \qquad (7)$$

Consider now the expression for Var $[p_i(n+1)|p_i]$. Using the form of $E[p_i(n+1)|p_i]$ from Theorem I, we obtain,

$$\text{Var}[p_i(n+1)|p_i] = E[p_i^2(n+1)|p_i] - E^2[p_i(n+1)|p_i]$$
$$= [1-(1-a^2)s_i]\, p_i^2 - [1-(1-a)]^2 p_i^2$$
$$= (1-a)^2\, s_i(1-s_i)p_i^2 \ .$$

Taking expectations again and using (7), we get

$$\mathrm{Var}[p_i(n+1)] = (1-a)^2 \, s_i(1-s_i) \, [1-(1-a^2)s_i]^n$$

which monotonically decreases with n and has a limit of zero as $n \to \infty$. ***

Remark: The fact that the limiting value of $\mathrm{Var}[p_i(n)]$ is zero is obvious from Theorem I, since $p_i(n)$ is a nonnegative random variable whose limiting mean is zero. However, the fact that it monotonically decreases to this limiting value is not necessarily obvious, and this is the main contribution of Theorem II.

Summarizing the results of the above theorems we note that given two elements R_j and R_k, the probabilities p_j and p_k converge w.p.1 to zero. Further, at any instant,

$$E[p_j(n)] > E[p_k(n)] \text{ if and only if } s_j < s_k \, .$$

Observe that this is true for all $0<a<1$. By appropriately choosing the value of the updating constant 'a', we shall now show that a stochastically stronger inequality exists - which not merely relates the expected values of p_j and p_k but the probabilities themselves.

Theorem III

For all j, k where $j \neq k$, if $s_j > s_k$, then, the quantity $\Pr[p_j(n) < p_k(n)]$ can be made as close to unity as desired.

Proof: Assume with no loss of generality that $s_j > s_k$. Let,

$$x_{j,k}(n) = p_j(n) \, / \, p_k(n) \, .$$

Clealy, $x_{j,k}(0)=1$. Further, $x_{j,k}(n)$ can only assume non-negative values. Consider the distribution of $x_{j,k}(n+1)$ given the values of $p_j(n)$ and $p_k(n)$. By virtue of (2),

$$\begin{aligned} x_{j,k}(n+1) &= (a\ p_j)\ /\ p_k && \text{w. prob. } s_j, \\ &= p_j\ /\ (a\ p_k) && \text{w. prob. } s_k, \\ &= p_j\ /\ p_k && \text{w. prob. } (1-s_j-s_k). \end{aligned}$$

$$\begin{aligned} \text{Thus, } x_{j,k}(n+1) &= a\ x_{j,k}(n) && \text{w. prob. } s_j, \\ &= x_{j,k}(n)\ /\ a && \text{w. prob. } s_k, \\ &= x_{j,k}(n) && \text{w. prob. } (1-s_j-s_k). \end{aligned} \qquad (8)$$

Taking conditional expectations yields,

$$E[x_{j,k}(n+1)\,|\,x_{j,k}(n)] = 1/a\ [a^2 s_j + a(1-s_j-s_k)+s_k]\ x_{j,k}(n).$$

Whence on taking expectation again and observing that $x_{j,k}(0)=1$, we get,

$$E[x_{j,k}(n)] = (h_{j,k})^n, \text{ where } h_{j,k} = 1/a\ [a^2 s_j + a(1-s_j-s_k)+s_k]. \qquad (9)$$

Let a be any real number satisfying, $s_k < a\ s_j$. Then,

$$\begin{aligned} & s_k/a < s_j \\ \Rightarrow\ & s_k(1-a)/a < s_j(1-a) \\ \Rightarrow\ & s_k(1/a - 1) < s_j(1-a) \\ \Rightarrow\ & a s_j - s_j - s_k + (s_k/a) < 0 \\ \Rightarrow\ & h_{j,k} < 1. \end{aligned}$$

Thus $h_{j,k}$ can be made strictly less than unity by appropriately choosing 'a'. From (9), this implies that $E[x_{j,k}(n)]$ can be rendered monotonically <u>decreasing</u> with n, and further,

$$\lim_{n\to\infty} E[x_{j,k}(n)] = 0$$

But $x_{j,k}(n)$ is always nonnegative. This implies that

$$\lim_{n\to\infty} x_{j,k}(n) \to 0 \qquad \text{w. prob. 1}$$

Thus, $\lim \Pr[x_{j,k}(n) > 0] \to 0$ w. prob. 1,

and the result follows. ***

<u>Remark</u>: Observe that although the values of $\{s_i\}$ are unknown, the above theorem says that by making 'a' sufficiently close to unity the asymptotic value $\Pr[p_j(n) < p_k(n)]$ can be made as close to zero as desired.

IV. A Stochastic Move-to-Front Algorithm

Using the results of Theorems I-III we shall now propose a stochastic move-to-front algorithm which is optimal for <u>all</u> distributions of access probabilities.

Initially, $p_i(0)$ is set to unity for all i. As each request is processed, the accessed element is moved to the front of the list with a probability $p_i(n)$ and the value of $p_i(n+1)$ is computed. After a sufficiently long sequence of accesses, the elements are sorted in the ascending order of the probabilities $p_i(n)$. Note that by this time the quantities $p_i(n)$ will be sufficiently small and thus the probability of getting out of this arrangement will be correspondingly small.

Rather than wait for a long period of time we propose to sort the list of elements repeatedly after every T accesses. T is called the Periodicity of Sorting, and a sequence of T consecutive accesses is called a Cycle. Initially the list is completely unsorted and so in O(N log N) time the elements can be arranged in the ascending order of $p_i(n)$. However, as the experimental results of the next section prove, after the list has been sorted a few times, the list remains "almost sorted" subsequently. This is because of the fact that although the p_i's do change subsequently, the number of move to the front operations performed diminish considerably.

Since the list remains "almost sorted", a sorting algorithm which performs well in such an environment should be used to perform the sorting. Dijkstra's Smoothsort [5] lends itself ideally to such an application. The latter is an excellent sorting algorithm which sorts a (nearly) sorted list in O(N) time and a completely unsorted list in O(N log N) time. Further, there is a smooth transition in the complexity of the algorithm as the number of unsorted elements increases. The hybrid algorithm proposed by Cook and Kim [4] could also be used to perform the sorting. Observe that the work done to sort the list is ultimately marginal - since it takes O(N) time to access the element at the tail of the list - which is the order of the time taken to sort the "almost sorted" list [5].

For the sake of completeness we now present algorithmically the list organizing scheme which uses stochastic Move-to-Front operations.

Algorithm I

Input: The updating constant 'a' and the Periodicity of Sorting T**

Output: A list which is dynamically reorganized using stochastic Move-to-Front operations.

Method: Initialize $p_i=1$ for i=1 to N

Repeat

for a sequence of T requests

move accessed element R_i to the front with probability p_i

$p_i=ap_i$

endfor

Smoothsort the list in ascending order of $\{p_j\}$

forever

Remark: With regard to computation, Algorithm I is more efficient than any of the algorithms known to us. This is because, as stated earlier, an access involves the modification of only one probability (which is the probability p_i associated with the accessed element R_i). Besides this, because the list operations are stochastic, a list operation is not necessarily performed on every access. Further, since the markovian representation of the scheme is absorbing, the number of list operations performed will asymptotically decrease to zero.

** It is possible to modify the algorithm so that the subsequent calls to the sorting routine are less frequent as the algorithm converges. This can be achieved by increasing T systematically every time the sorting routine is called. However, the asymptotic properties of either arrangement would be the same.

IV. Experimental Results

The list organizing scheme proposed in this paper was tested for files with various numbers of records and three types of distributions defined below as DIST1, DIST2, and DIST3.

DIST1: $s_i = k_1.(N-i+1)$, where, $k_1 = 1/(\sum_{i=1}^{N} i)$.

DIST2: $s_i = k_2 / i$, where, $k_2 = 1/(\sum_{i=1}^{N} 1/i)$.

DIST3: $s_i = k_3 / (2^i)$, where, $k_3 = 1/(\sum_{i=1}^{N} 1/2^i)$.

These distributions have been extensively studied in the literature [11]. Note that DIST2 is the familiar Zipf's law.

Ten experiments, each consisting of 4000 requests were performed on lists of various sizes. The periodicity of sorting, was 100. As a measure of the performance of the scheme, the accuracy of the scheme was computed in terms of the number of distinct pairs which were correctly organized. We refer to this measure as ACC, where,

$$ACC = \frac{\text{No. of distinct pairs correctly organized}}{\binom{N}{2}}$$

To measure of the convergence characteristics and the process of stabilization of the organization of the list, we have also evaluated a quantity, K_T, which is the number of moves to the front that are performed per cycle. In other words, K_T is the number of times the list is manipulated between calls to the sorting routine.

Plots of the average values of ACC and K_T for N=10 and 40 are given in Figures I and II respectively. The distribution of

the access probabilities in these figures obeyed Zipf's law. In the case when the number of records was 10, the initial average value of ACC was 0.55. After the first sort it rose to 0.93, and within the next twelve cycles the average value of ACC settled at its terminal value of 0.96. The average value of K_T was initially 29.50. After the first sort it fell to 5.10 and then, before the next 5 cycles, its value fell to less than one. This means that on the average, the list was manipulated less than once per cycle after 6 calls to the sorting routine.

Simulation results for N=10,15,25 and 40 and for all three types of distributions are given in Table I. These results clearly demonstrate the power of the scheme we have proposed.

N	Type of Distribution	$\hat{ACC}(0)$	$\hat{ACC}(10)$	$\hat{K}_T(0)$	$\hat{K}_T(10)$
10	DIST1	0.48	0.92	35.10	0.30
	DIST2	0.55	0.94	29.50	0.40
	DIST3	0.41	0.90	18.20	0.30
15	DIST1	0.50	0.89	43.00	0.70
	DIST2	0.50	0.90	33.80	1.00
	DIST3	0.51	0.83	21.50	0.20
25	DIST1	0.52	0.87	52.00	1.00
	DIST2	0.50	0.88	36.50	1.10
	DIST3	0.49	0.83	18.60	0.40
40	DIST1	0.52	0.87	60.80	0.90
	DIST2	0.48	0.87	37.00	1.40
	DIST3	0.46	0.87	19.70	0.40

TABLE I: Simulation Results for the Stochastic Move-to-Front List Organizing Scheme

Notation:

N: No. of elements in the list.

$\hat{ACC}(k)$: Average value of the percentage of the number of pairs which are correctly organized after the kth sort.

$\hat{K}_T(k)$: Average value of the number of move to the front operations performed in the kth cycle.

DIST1: $s_i = k_1 . (N-i+1)$, where, $k_1 = 1/(\sum_{i=1}^{N} i)$.

DIST2: $s_i = k_2 / i$, where, $k_2 = 1/(\sum_{i=1}^{N} 1/i)$.

DIST3: $s_i = k_3 / (2^i)$, where, $k_3 = 1/(\sum_{i=1}^{N} 1/2^i)$.

IV. Conclusions

We have considered a list of elements $\{R_1,\ldots,R_N\}$ in which the element R_i is accessed with a probability s_i. The latter is unknown apriori. A list organizing scheme has been proposed. The scheme performs a move to the front operation on the accessed element R_j with a probability p_j which is systematically decreased. Ultimately, the list gets absorbed into one of the N! possible arrangements. We have shown that the asymptotic probability of converging to the optimal arrangement can be made as close to unity as desired.

Acknowledgements

I am greatly indebted to my colleague Nicola Santoro who introduced me to the problem and to the literature in the field. I am grateful to him for his encouragement and comments during the course of the study. I would also like to acknowledge my gratitude to Jorg Sack and Thomas Strothotte who informed me of Dijkstra's Smoothsort algorithm. Finally, I would like to thank my colleague Michael Atkinson for his critical reading of the manuscript and his comments which rendered the paper more readable.

References

[1] Arnow, D.M. and Tenebaum, A.M., "An Investigation of the Move-Ahead-k Rules", Congressus Numerantium, Proc. of the Thirteenth Southeastern Conference on Combinatorics, Graph Theory and Computing, Florida, February 1982, pp.47-65.

[2] Bitner, J.R., "Heurishics That Dynamically Organize Data Structures", SIAM J. Comput., Vol.8, 1979, pp.82-110.

[3] Burville, P.J. and Kingman, J.F.C., "On a Model for Storage and Search", J. Appl. Probability, Vol.10, 1973, pp.697-701.

[4] Cook, C.R., and Kim, D.J., "Best Sorting Algorithm for Nearly Sorted Lists", Comm. ACM, Vol.23, 1980, pp.620-624.

[5] Dijkstra, E.W., "Smoothsort, An Algorithm for Sorting in SITU", Science of Computer Programming, 1982, pp.223-233.

[6] Gonnet, G.H., Munro, J.I. and Suwanda, H., "Exegesis of Self Organizing Linear Search", SIAM J. Comput., Vol. 10, 1981, pp.613-637.

[7] Hendricks, W.J., "The Stationary Disribution of an Interesting Markov Chain", J. App. Probability, Vol.9, 1972, pp.231-233.

[8] Hendricks, W.J., "An Extension of a Theorem Concerning an Interesting Markov Chain", J. App. Probability, Vol.10, 1973, pp.231-233.

[9] Kan, Y.C. and Ross, S.M., "Optimal List Order Under Partial Memory Constraints", J. App. Probability, Vol.17, 1980, 1004-1015.

[10] Karlin, S. and Taylor, H.M., "A First Course in Stochastic Processes", Academic Press, 1975.

[11] Knuth, D.E., "The Art of Computer Programming, Vol.3, Sorting and Searching", Addison-Wesley, Reading, Ma, 1973.

[12] McCabe, J., "On Serial Files With Relocatable Records", Operations Research, Vol.12, 1965, pp.609-618.

[13] Rivest, R.L., "On Self-Organizing Sequential Search Heuristics", Comm. ACM, Vol.19, 1976, pp.63-67.

[14] Sleator, D. and Tarjan, R., "Amortized Efficiency of List Update Rules", Proc. of the Sixteenth Annual ACM Symposium on Theory of Computing, April 1984, pp.488-492.

[15] Tenenbaum, A.M. and Nemes, R.M., "Two Spectra of Self-Organizing Sequential Search Algorithms", SIAM J. Comput., Vol.11, 1982, pp.557-566.

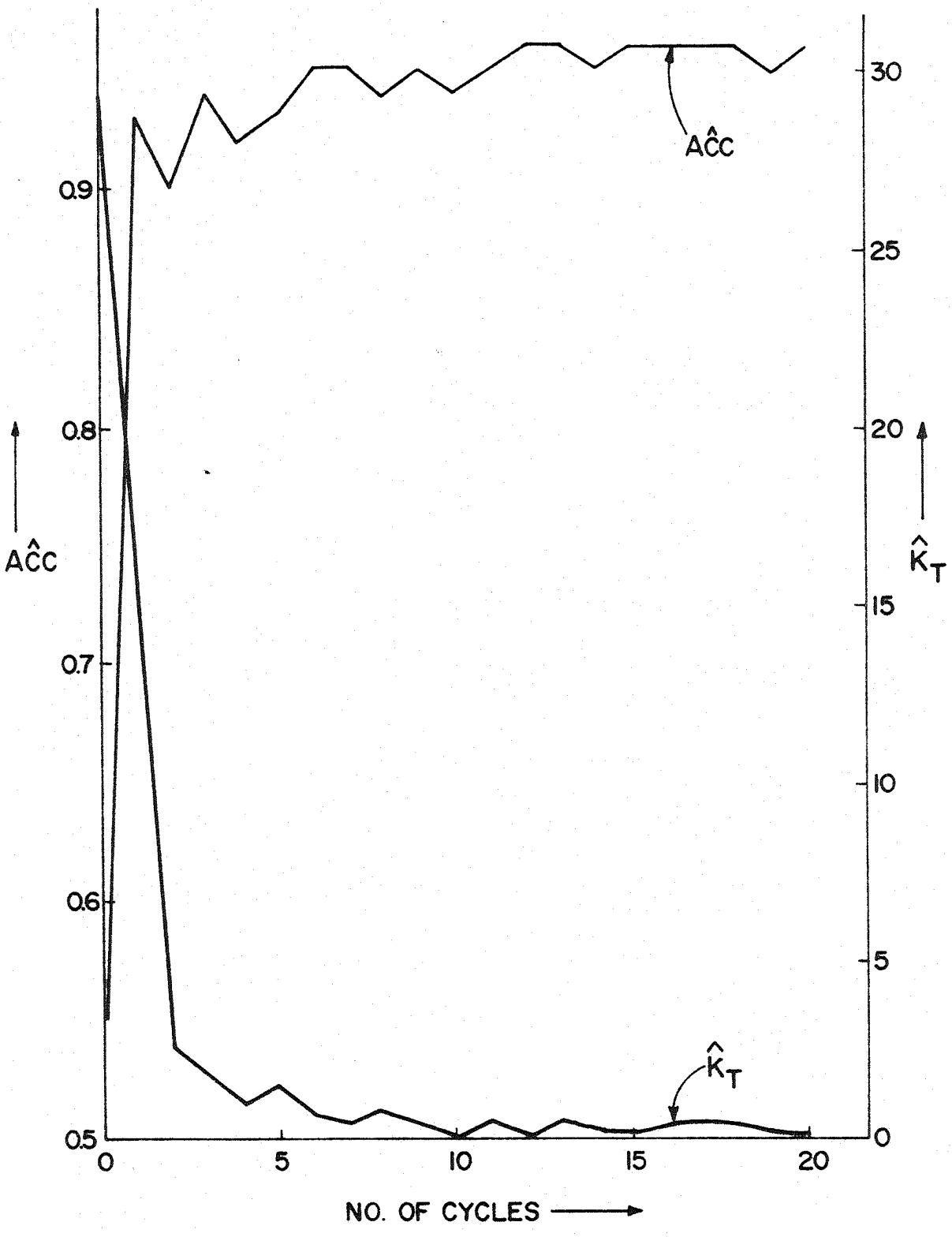

Figure 1: Variation of $\hat{ACC}$ and $\hat{K}_T$ with time for N = 10.
The parameters of the scheme are T=100 and a =0.9.
The Access Probabilities obey Zipf's Law.

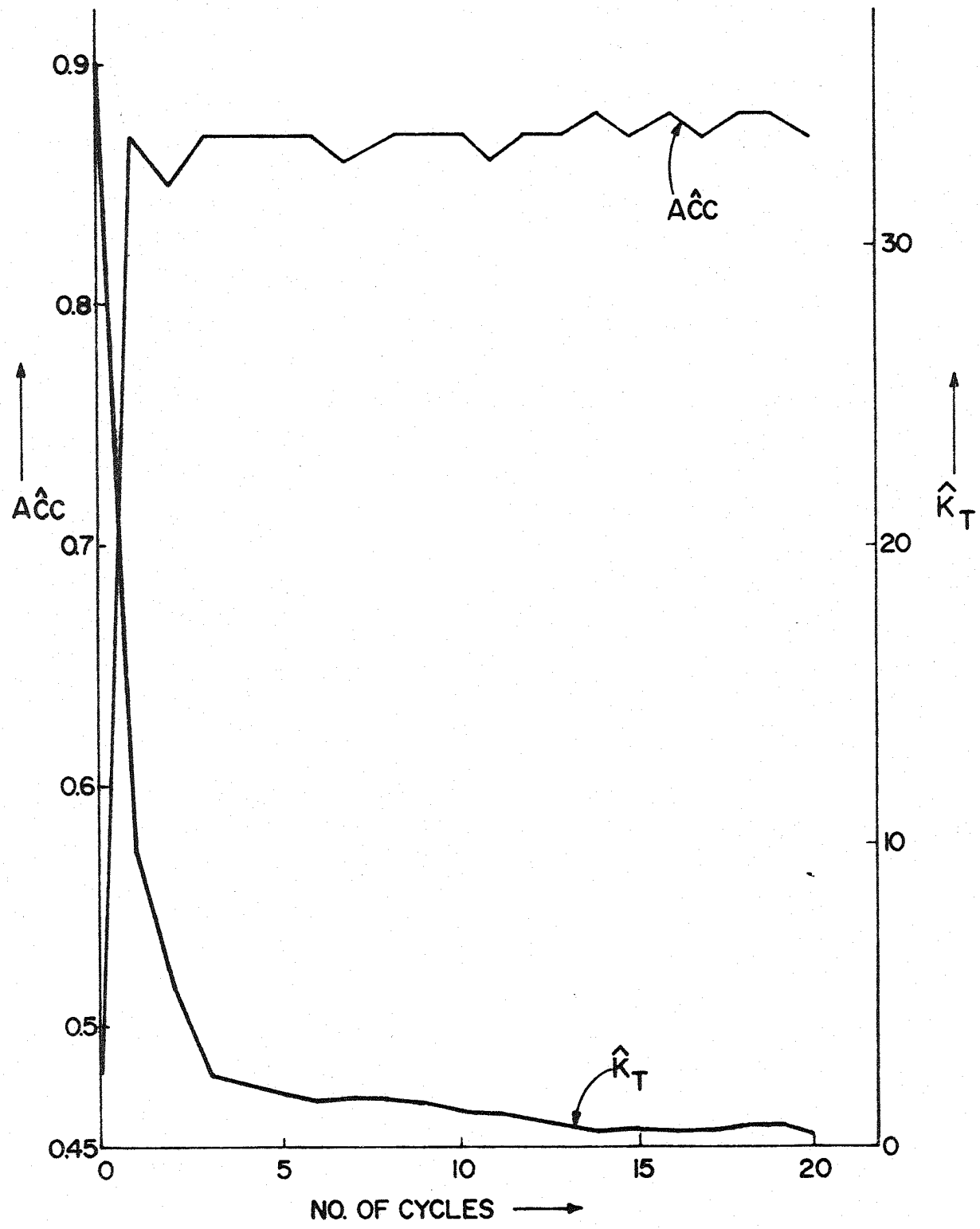

Figure II : Variation of $\hat{ACC}$ and $\hat{K}_T$ with time for N = 40.
The parameters of the scheme are T = 100 and a = 0.9.
The Access Probabilities obey Zipf's Law.