SELF-ORGANIZING LISTS AND INDEPENDENT REFERENCES — A STATISTICAL SYNERGY

Micha Hofri† and Hadas Shachnai

Computer Science Department, Technion - Israel Institute of Technology
Haifa 32000, Israel

(October 1988, revised April 1997).

ABSTRACT

Let $R_1, \cdots, R_n$ be a linear list of n elements. We assume the independent reference model, with a fixed but unknown access probability vector. We survey briefly the problem of reorganizing the list dynamically, on the basis of accrued references, with the objective of minimizing the expected access-cost. The *Counter Scheme (CS)* is known to be asymptotically optimal for this purpose.
The paper explores the *CS*, with the aim of reducing its storage requirements. We start with a detailed exposition of its cost function and then point out that it interacts with the access model to produce some remarkable synergistic effects. These make it possible to use very effective 'truncated versions' of the *CS*, which have very modest space requirements. The versions we consider are:
(i) The 'Limited-Counters Scheme', which bounds each of the frequency counters to a maximal value c.
(ii) The original *CS* with a bound on the number of references during which the scheme is active. The bound is chosen to achieve a desired access cost, compared with the optimal policy.

1. Introduction

A linear list is a data structure in which there is little structure. Thus it is simple to build and maintain, with the consequent penalty that it is relatively inefficient in use. Nevertheless, under quite common circumstances it is the structure of choice. We assume the following layout:

A list L contains n records $\{R_i, 1 \le i \le n\}$, that are identified uniquely by their keys $\{K_i\}$. The content of the list is assumed to remain unchanged over time.

The list structure implies a sequential search *access scheme:* in order to access R_i, the keys in positions 1, 2, $\cdots$ are compared successively against K_i until an equality is obtained. The number of comparisons is defined as the *cost* of the access, and denoted by $\boldsymbol{C}$. The following *reference model* is assumed: each request may address the record R_i for $1 \le i \le n$, with a time-homogeneous probability $p_i > 0$. This holds true independently of the list order and the history of past accesses. Such a scheme is known as the (stationary) independent-reference model (*irm*).

A natural objective is to have the smallest possible cost per access. If the reference-probability vector (*rpv*) $\boldsymbol{p} = \{ p_i \}$ were known, the optimal policy would be to keep the list sorted by p_i , in non-increasing *static* order. The term "self-organizing" in the title bespeaks of the situation where the *rpv* is *not known*, and the access mechanism activates periodically – typically following each access – a procedure that may change the order of the records, based on past

†Currently with the Department of Computer Science, Rice University, Houston TX 77005-1892.

references (or any other criterion).

Let $\sigma(i)$ denote the location of R_i in the list. The state of the list is then the permutation σ. In terms of this notation the expected cost may be written as $C \equiv E[\boldsymbol{C}] = E[\sigma(I)]$, where I is the identity of the accessed record.

Various policies have been proposed to minimize C, using the reference history. Rivest (1976) calls them "permutation rules". Hester and Hirschberg (1985) survey most of the work in this area published by 1983. We describe three of the better known reorganization-schemes. They are activated following each reference.

Move to the front (MTF): Following an access to R_i, in position $\sigma(i)$, it is placed at the head of the list, pushing the records in locations $1, 2, \cdots, \sigma(i)-1$ one step back. This policy has been studied extensively. Two measures of its operation were examined. Its asymptotic cost per access under various *rpv*'s has been discussed by Bitner (1979), Burville and Kingman (1973), Hendricks (1976), Knuth (1973), McCabe (1965) and Rivest (1976). They show that the limiting expected cost per access is given by

$$C(MTF|\boldsymbol{p}) = 1 + \sum_{1\le i<j\le n} \frac{2p_i p_j}{p_i + p_j}\,. \tag{1.1}$$

Bitner (1979) considered also the cost of requests while the list state converges to its asymptotic value. More precisely, he defined the "overwork" (OW) for any policy H as

$$OW_H \equiv \sum_{m\ge 0} [C_m(H|\boldsymbol{p}) - C(H|\boldsymbol{p})], \tag{1.2}$$

where $C_m(H|\boldsymbol{p})$ is the expected cost of the $m+1$st access, and $C(H|\boldsymbol{p}) = \lim_{m\to\infty} C_m(H|\boldsymbol{p})$. The expectation is evaluated with respect to the initial state (which is assumed to be uniformly distributed) and the history of references. Note that in general OW_H need not even be finite for all H; however, in all the cases we shall examine, the list reorganization process can be described by a first-order Markov chain, and then OW_H is guaranteed to be finite. For one reason or another, MTF is the policy that has been studied most, and is commonly used as a yardstick by which other rules are measured. We shall follow this usage to some extent.

Recently, there have appeared some work [Sleator and Tarjan (1985), Bentley and McGeogh (1985)] that considers not the *expected* cost of MTF, but rather the highest possible cost (worst case), when averaged (or – as called in that context – amortized) over a long reference sequence. Since the *irm* is irrelevant there, this approach is outside the scope of the present paper.

Transposition rule (TR): A record R_i, when accessed at location $\sigma(i) > 1$, is transposed with its neighbor in location $\sigma(i)-1$, getting thus one step closer to the head of the list. This rule was found to be more efficient asymptotically than the MTF scheme [Hendricks, (1976), Rivest (1976)]. However, the rate of convergence to its desirable limiting cost is known to be lower [Bitner (1979), Hester and Hirschberg (1985)].

Counter Scheme (CS): Each record R_i is associated with a frequency counter c_i, which is incremented every time the record is requested. The record is then moved forward (if needed) to keep the records sorted in a non-increasing order of their counters.

Let m be the number of requests made to the list; from the (strong) law of large numbers,

$$\frac{c_i}{m} \underset{m\to\infty}{\to} p_i\,, \qquad 1 \le i \le n \quad \text{(with prob. 1)}.$$

This implies that in the limit the list is optimally ordered. This advantage over other policies is not only a property of the limiting cost: a first step in showing that was made by Lam *et al.* (1981). They consider a slightly generalized *CS* (of which the above *CS* is a special case; they allow the initial state of the list, when the counters are reset, to depend on the operation of some other permutation rule up to that time). They then show that keeping the records in sorted order

of their counters is better than any other *counter-based* rule, at *every stage*. We recently proved a stronger optimality claim for the *CS*, in Hofri and Shachnai (1989).

In spite of its desirable cost-function evolution, the *CS* has not attracted much attention in the published literature. The reason is real enough – compared with MTF, TR and similar varieties, it is the only one requiring the maintenance of additional information – the counters c_i. Also, when compared with other rules that maintain some administrative records, the *CS* has been denigrated for having "infinite space complexity", since the counters require an unbounded amount of storage space, when the rule is implemented indefinitely. Possibly it is the claim of *limiting* optimality that has caused this attitude.

Bitner (1979) suggested a few variations on the *CS* in order to overcome this problem, and experimented with a few simple $\boldsymbol{p}$. First, he considered the limited-difference rule, which stores the differences between successive counters and, in addition, imposes an upper bound on the size of the stored values. This version of the *CS* does not converge to the optimal ordering. However, as the upper bound on the difference fields is increased (and so does their space requirement) the ultimate ordering improves, and its asymptotic access-cost approaches the minimum. Secondly, he examined two classes of rules which are combinations of *CS* with other permutation rules. In 'wait c, move and clear', each record R_i has a counter c_i which is initially set to zero. When a counter c_i obtains the value c, the record R_i is moved according to the specified permutation rule (MTF, TR, etc.) and all the counters are reset. The class of 'wait c and move' rules is the same except that when a record R_i is moved, only c_i is reset. Both classes improve the asymptotic access-costs of the original rules, but their asymptotic cost and rate of convergence is poorer than that of the limited-difference rule.

The latter two modifications may be viewed as an attempt to correct the main shortcoming of MTF: when a "rare" record (one having a low p_i) is referenced, it is moved all the way to the front, and will take a long time to dribble back to an inexpensive position.

Another permutation rule that maintains additional information was suggested in Oommen and Hansen (1987). They considered the seemingly quaint "stochastic move to the rear" scheme, which is reminiscent of simulated annealing. In this scheme an accessed record R_j may be moved to the *rear* of the list, with probability q_j. The value of q_j decreases on each access. Eventually, as the q_j's become negligible, the list reaches a steady state, in the sense that the records tend to stay in place. Since records with low p_i are accessed less frequently, they are the ones expected to keep moving at later times, and land at the rear of the list. With the proposed scheme $q_j = q(c_j) = a^{c_j}$ (with some $0 < a < 1$), keeping track of the counters is sufficient to support the reorganization. The upper limit on the size of counter fields can be determined by the assumption that the q_j's cannot get smaller than some machine-dependent constant. Hence the operational and storage overhead is like that of a limited-counters CS.

The authors point out that their scheme does better than the *CS* with respect to pairs of records with *equal* access probabilities: with *CS*, these records will keep being exchanged indefinitely, since the event {the counters of two records with equal probabilities are equal} is recurrent. Strictly speaking, however, it is null-recurrent: the expected number of references (to these records) between successive occurrences of this event is infinite.

A disadvantage of this method is its large overwork, as defined in equation (1.2). Clearly, the greater part of the time the scheme is active, before settling down, it spends on locating properly those records with the smaller p_i's. Those probabilities contribute little to the expected access-cost and therefore need not be known too precisely, or sorted with much circumspection: it is enough if they are at the rear of the list and "out of the way" most of the time. With this scheme, however, both during the initial phase, when popular records still land often at the rear, and later, when 'rare' records are still likely to be too close to the beginning of the list, the expected cost is

quite high.

The last point touches on the main thesis of this paper: a reorganization rule should estimate the correct order of the important records, and do it as promptly as possible. Hence the appeal of the *CS*: recall that frequencies are sufficient statistics for the estimation of the *rpv* of a multinomial process. Moreover, they are (the most) efficient ones. There is a substantial statistical literature on discriminating multinomial probabilities, under various requirements; a comprehensive account is Gibbons *et al.* (1977). Here, however, the requirement is merely to minimize the expected value $E[\sigma(I)]$, where I is the index of the accessed record. The *irm* and this cost function interact and give rise to the following synergistic effects:

– Broadly speaking, only the records with large probabilities make any significant contribution to the cost function. However, it is precisely those records for which the counters increase fast and provide promptly good estimates.

– Actually it is not even the probabilities proper we need to estimate, but only their sorted order. When the difference between two probabilities, say $p_i - p_j > 0$ is substantial, then after a relatively short reference sequence there will be a very small probability for the event $\{c_i < c_j\}$. On the other hand, when $p_i - p_j > 0$ is very small, even though the number of references required to order them correctly with high probability is huge [Gibbons *et al.* (1977)], the corresponding penalty of incorrect order is minute (even when p_i and p_j proper are *not* small – since the added cost is proportional to $p_i - p_j$). We conclude that the *CS* is well-positioned to utilize this synergy.

The rest of the paper justifies and quantifies in various ways these claims, and outlines efficient procedures to take advantage of their promise.

We present two ways for limiting the space required by *CS* without damaging its superior performance.

One idea is to fix a maximal value $c_{\max}$ which the counters may not exceed. The other method stops the list-reorganization process after a finite, predetermined number of requests. This approach not only bounds the required space, but also sets a limit on the time required to implement the counter scheme. No such bound has been presented in previous work, for any permutation rule. Limiting the activation period is attractive in itself, by reducing the overhead involved in the usage of these policies.

In Section 2 the expected access-cost function using *CS* is computed, shown to be monotonic in m, the number of references, and that it converges to the expected access-cost of the optimal ordering as $m \to \infty$.

In Section 3 we examine this convergence in closer detail, and show that for any $\alpha > 0$ the expected access-cost gets to within a factor of $1 + \alpha$ of the minimum (optimal) expected cost within a finite number of requests − that we can bound. We derive two estimates for the required number of references. The first depends explicitly on the values of all access probabilities. We then exhibit a "non-parametric" bound: while it depends on the values of α and n, it does *not* depend on the *rpv* $\boldsymbol{p}$.

In Section 4 we present a variation of *CS* which limits the space allocated to the counter fields by limiting their size to $c_i \le c_{\max}$, where $c_{\max}$ may be any positive integer. Its performance is shown to compare surprisingly well with the optimal static ordering, even for quite small $c_{\max}$. Indeed, we show that even for $c_{\max} = 1$, the expected access-cost for each request is as good as that of the MTF rule, which is considerably more expensive to implement.

We conclude with a discussion of the assumptions of the model and point out that there are many questions concerning it, and its immediate extensions, that are as yet unanswered.

2. Definitions and Preliminary Results

The measure we use to compare the *CS* to other policies (and, in particular, to the optimal static ordering) is C, the expected access-cost at each stage. Let $\boldsymbol{p} = (p_1, \cdots, p_n)$ be the reference-probabilities vector (*rpv*). We use $C_m(H|\boldsymbol{p})$ to denote the expected cost to access the list under the scheme H for the $m+1$st reference, $m \ge 0$. Without loss of generality, we may assume a renumbering such that $p_1 \ge p_2 \ge \cdots \ge p_n$. With this enumeration, the expected access cost under the optimal static order is given by

$$C_m(OPT|\boldsymbol{p}) = C(OPT|\boldsymbol{p}) = \sum_{i=1}^{n} i p_i.$$

Lemma 2.1: If the initial order of the list is random, with equal probability for each permutation of L, then

$$C_m(CS|\boldsymbol{p}) = 1 + \sum_{1 \le i < j \le n} E_m(p_i, p_j), \tag{2.1}$$

where

$$E_m(p_i, p_j) \equiv \sum_{k=0}^{m} \binom{m}{k} (1 - p_i - p_j)^{m-k} E(k; p_i, p_j). \tag{2.2}$$

and

$$E(k; p_i, p_j) \equiv p_i \sum_{r=\lfloor k/2 \rfloor + 1}^{k} \binom{k}{r} p_j^r p_i^{k-r} + \frac{p_i + p_j}{2} I_k \cdot \binom{k}{k/2} (p_i p_j)^{k/2} + p_j \sum_{r=\lfloor k/2 \rfloor + 1}^{k} \binom{k}{r} p_i^r p_j^{k-r}. \tag{2.3}$$

where an empty sum on r vanishes, and $I_k = \begin{cases} 1 & \text{when } k \text{ is even} \\ 0 & \text{otherwise.} \end{cases}$

Proof: The cost of accessing a record can be obtained by summing over pairs of records, with the contribution of the pair (R_i, R_j) is the number of *unsuccessful* comparisons of K_i and K_j denoted by $E_m(p_i, p_j)$; hence equation (2.1). This value is obtained by conditioning on k, the number of times these two records have been referenced; the conditional cost is given by equation (2.2), when we define

$$E(k; p_i, p_j)/(p_i + p_j)^k = p_i \cdot \text{Prob}(R_j \text{ precedes } R_i \,|\, k) + p_j \cdot \text{Prob}(R_i \text{ precedes } R_j \,|\, k), \tag{2.4}$$

and the right-hand side of (2.4) is precisely the value given in equation (2.3). □

The following property of *CS* is intuitively obvious; the proof requires however a rather detailed analysis of $C_m(CS|\boldsymbol{p})$. Some of the intermediate results will serve us later.

Observation 2.2: (Monotonicity) The expected cost $C_m(CS|\boldsymbol{p})$ is monotonically decreasing in m, i.e.

$$C_m(CS|\boldsymbol{p}) \le C_{m-1}(CS|\boldsymbol{p}) \quad \text{for all } m \ge 1. \tag{2.5}$$

Proof: Equation (2.1) implies that it suffices to show for arbitrary $1 > p_i \ge p_j > 0$ and all $m \ge 1$ that

$$E_{m-1}(p_i, p_j) - E_m(p_i, p_j) \ge 0.$$

We first show that the $E(k; p_i, p_j)$ also satisfy a monotonicity property. Consider the case $p_i + p_j = 1$. We separate the cases of even and odd number of references to R_i and R_j:

(i) Let $k = 2l + 1$ for an integer $l \geq 0$, then

$$E(k; p_i, p_j) = p_i + (p_j - p_i) \sum_{r=l+1}^{2l+1} \binom{2l+1}{r} p_i^r p_j^{2l+1-r}, \tag{2.6}$$

and

$$E(k-1; p_i, p_j) = p_i + (p_j - p_i)\Big[\sum_{r=l+1}^{2l} \binom{2l}{r} p_i^r p_j^{2l-r} + \frac{1}{2}\binom{2l}{l} p_i^l p_j^l \Big]. \tag{2.7}$$

Taking the difference and expanding the coeffi cient $\binom{2l+1}{r}$, the sums cancel out to leave

$$E(k-1; p_i, p_j) - E(k; p_i, p_j) = \frac{1}{2}\binom{2l}{l} p_i^l p_j^l (p_j - p_i)^2 \ \geq 0. \tag{2.8}$$

(ii) When $k = 2l, l \geq 1$, the difference between the expressions equivalent to (2.6) and (2.7) can be similarly manipulated to show $E(k-1; p_i, p_j) - E(k; p_i, p_j) = 0$. Hence, for all $k \geq 1$, $0 < p_j \leq p_i < 1$, where $p_i + p_j = 1$, the expectations $E(k; p_i, p_j)$ are monotonic decreasing in k.

Consider now $p_i + p_j \leq 1$. From equation (2.3):

$$E(k; p_i, p_j) = (p_i + p_j)^{k+1} E(k; p_i/(p_i + p_j), p_j/(p_i + p_j)). \tag{2.9}$$

Then, from equation (2.2), the difference $E_{m-1}(p_i, p_j) - E_m(p_i, p_j)$ is given by

$$\begin{aligned} &\sum_{k=1}^{m-1} \binom{m-1}{k} (1 - p_i - p_j)^{m-1-k} E(k; p_i, p_j) + \frac{1}{2}(p_i + p_j)(1 - p_i - p_j)^{m-1} \\ &- \sum_{k=1}^{m} \binom{m}{k} (1 - p_i - p_j)^{m-k} E(k; p_i, p_j) - \frac{1}{2}(p_i + p_j)(1 - p_i - p_j)^m. \end{aligned} \tag{2.10}$$

Equation (2.9), and the steps that produced equation (2.8) reduce this to

$$\begin{aligned} &\sum_{k=1}^{m-1} \binom{m-1}{k} (1 - p_i - p_j)^{m-1-k} (p_i + p_j)^{k+2} [E(k; p_i/(p_i + p_j), p_j/(p_i + p_j)) \\ &\quad - E(k+1; p_i/(p_i + p_j), p_j/(p_i + p_j))] + \frac{1}{2}(1 - p_i - p_j)^{m-1} (p_i + p_j)^2, \end{aligned} \tag{2.11}$$

which is non-negative, due to the shown monotonicity. □

Consider now the limiting state, when the number of accesses m increases. The convergence of the list-state to the optimal order follows from the strong law of large numbers. It may be shown however directly in the cost function, by letting $m \to \infty$ in equations (2.2-3). This yields, for all $1 \geq p_i \geq p_j \geq 0$,

$$E_m(p_i, p_j) \underset{m \to \infty}{\to} p_j. \tag{2.12}$$

Substituting this limit in equation (2.1) is enough in order to verify that

$$\lim_{m \to \infty} C_m(CS|\boldsymbol{p}) = \sum_{i=1}^{n} i p_i = C(OPT|\boldsymbol{p}). \tag{2.13}$$

3. A Stopping Point for the Counter-Scheme

The literature about self-organizing lists assumes tacitly that any reorganizing rule is to be implemented indefi nitely; sometimes the woolly phrase "till convergence to a steady state" is preferred. For the *CS* this is impractical and unnecessary. After a predictable number of requests, the cost function is already "close enough" to the optimal cost, and any subsequent updates serve only to reduce it by insignifi cant amounts. This property, which holds for any *rpv* is the essence

of the synergy discussed in Section 1. One can argue intuitively for it as follows: The utility of *CS* is apparent for two broad categories of *rpv*s. For a 'steep' distribution function – one in which there is a substantial variation between the p_is – the records with higher access probabilities are the first ones to organize properly, at the front, whereas the records which are less frequently accessed remain longer unordered in the rear of the list. But the latter contribute—by virtue of being rarely referenced—much less to the expected access-cost. Therefore, with high probability $C_m(CS|\boldsymbol{p})$ reaches a low value at an early stage, even though only a small part of the records are expected to be ordered correctly by then. On the other hand, when the distribution function is flat, the rate of convergence to the optimal ordering is slow (due to the hard-to-detect slight differences between the access probabilities); but then, *any* ordering, including the optimal one, would provide a similar, relatively high, expected cost. Again, implementing the scheme indefinitely would not improve much the expected access-cost computed already nearly at the beginning. The extreme instance is the uniform *rpv* $\bar{\boldsymbol{p}}$, with $p_1 = p_2 = \cdots = p_n = 1/n$. In that case, nothing helps (or matters), and

$$C_m(H|\bar{\boldsymbol{p}}) = C(OPT|\bar{\boldsymbol{p}}) = \frac{n+1}{2}\,,$$

for any rule H at any stage.

The following result suggests a procedure for deriving a stopping point, in the hypothetical scenario where the *rpv* is known.

Lemma 3.1: Let $\boldsymbol{p}$ be a *rpv*, relabeled so that $1 > p_1 \geq p_2 \cdots \geq p_n > 0$, with $n > 2$. Then,

$$C_m(CS|\boldsymbol{p}) \;\leq\; (1+\alpha)\cdot C(OPT|\boldsymbol{p}) \;\text{ for all }\; m \geq m^*, \tag{3.1}$$

where m^* is the integer part of

$$\frac{\sum\limits_{i,j\in S} \dfrac{p_i + p_j}{2\,(p_i - p_j)}}{\alpha \sum\limits_{i,j\in S}\Big(p_j + \dfrac{2}{n(n-1)}\Big)}, \tag{3.2}$$

and where

$$S = \{1 \leq i < j \leq n :\; p_i - p_j > 2t \equiv 2\alpha\Big(p_j + \frac{2}{n(n-1)}\Big)\}. \tag{3.3}$$

Proof: We need to show that $\Delta^{(m)} \equiv C_m(CS|\boldsymbol{p}) - (1+\alpha)C(OPT|\boldsymbol{p}) \leq 0$ for all $m \geq m^*$. Using equations (2.1) and (2.13) we write

$$\Delta^{(m)} = 1 + \sum_{1\leq i<j\leq n} [E_m(p_i, p_j) - (1+\alpha)p_j] - (1+\alpha) \tag{3.4}$$

We simplify the relation by observing that $\Sigma_{1\leq i<j\leq n} 2/n(n-1) = 1$, to bring the whole of the right-hand side of (3.4) under the summation sign

$$\Delta^{(m)} = \sum_{1\leq i<j\leq n} \Delta^{(m)}(p_i, p_j) \equiv \sum_{1\leq i<j\leq n} \Big[E_m(p_i, p_j) - \frac{2\alpha}{n(n-1)} - (1+\alpha)p_j\Big]. \tag{3.5}$$

Comment: The choice of the term $2/(n(n-1))$ above was quite arbitrary. Actually we could specify any function $d_{i,j}$ such that $\Sigma_{1\leq i<j\leq n} d_{i,j} = 1$; then we could allocate these values among the pairs (i, j) so as to maximize the denominator of the expression for m^*. The improvement thus obtainable was found to be quite marginal.

Since we cannot obtain an explicit expression for the minimal value of m that makes the entire sum negative, we first consider each term *separately*. From Observation 2.2 it follows that $\Delta^{(m)}(p_i, p_j)$ is monotonic nonincreasing in m. Evaluating $\Delta^{(0)}(p_i, p_j)$ we find

$$\Delta^{(0)}(p_i, p_j) = \frac{p_i + p_j}{2} - \frac{2\alpha}{n(n-1)} - (1+\alpha)p_j. \tag{3.6}$$

Hence, the monotonicity guarantees that whenever $p_i - p_j \le 2t$, as defined in (3.3), there is nothing further to search for. The rest of the derivation assumes $p_i - p_j > 2t$. It would suffice to solve the equation $\Delta^{(m)}(p_i, p_j) = 0$ – except that we found no explicit solution for this equation either. Thus, we proceed as follows: define

$$G(x; p_i, p_j) = \sum_{m=0}^{x-1} \Delta^{(m)}(p_i, p_j). \tag{3.7}$$

Let x^* be a solution of the equation $G(x; p_i, p_j) = 0$, then obviously, $\Delta^{(m)}(p_i, p_j) \le 0$ for all $m \ge \lfloor x^* \rfloor$. It will prove simple to find an upper bound for $G(x; \cdots)$ for which the corresponding equation could be solved easily.

The recursion for $E(k; p_i, p_j)$ obtained in (2.8) and (2.9) provides for $k \ge 1$

$$\begin{aligned} E(k; p_i, p_j) &= E(1; p_i, p_j) - [E(1; p_i, p_j) - E(2; p_i, p_j)] - \cdots - [E(k-1; p_i, p_j) - E(k; p_i, p_j)] \\ &= (p_i + p_j)^{k-1}\Big[2p_i p_j - \tfrac{1}{2}(p_i - p_j)^2 \sum_{r=1}^{k_1} \binom{2r}{r} \Big(\frac{p_i}{p_i + p_j}\Big)^r \Big(\frac{p_j}{p_i + p_j}\Big)^r\Big], \quad k_1 \equiv \Big\lfloor \frac{k-1}{2} \Big\rfloor. \end{aligned} \tag{3.8}$$

From (2.3) follows that $E(0; p_i, p_j) = (p_i + p_j)/2$. Substituting into equations (2.2) and (3.5), the identity $\Sigma_{r\ge 0}\binom{2r}{r}[p_i p_j/(p_i + p_j)^2]^r = (p_i + p_j)/(p_i - p_j)$ gives

$$\begin{aligned} \Delta^{(m)}(p_i, p_j) = \tfrac{1}{2}\Big(\frac{p_i - p_j}{p_i + p_j}\Big)^2 \Big[(1 - p_i - p_j)^m + \sum_{k=0}^{m} \binom{m}{k} (1 - p_i - p_j)^{m-k} (p_i + p_j)^{k+1} \\ \times \sum_{r > k_1} \binom{2r}{r} \Big(\frac{p_i}{p_i + p_j}\Big)^r \Big(\frac{p_j}{p_i + p_j}\Big)^r\Big] - \alpha p_j - \frac{2\alpha}{n(n-1)}. \end{aligned} \tag{3.9}$$

The feasibility of this proof stems from the obliging manner equation (3.9) separates terms that do or do not depend on m. Introducing this into equation (3.7):

$$\begin{aligned} G(x; p_i, p_j) = \tfrac{1}{2}\Big(\frac{p_i - p_j}{p_i + p_j}\Big)^2 \sum_{m=0}^{x-1} \Big[(1 - p_i - p_j)^m + \sum_{k=0}^{m} \binom{m}{k} (1 - p_i - p_j)^{m-k} (p_i + p_j)^{k+1} \\ \times \sum_{r > k_1} \binom{2r}{r} \Big(\frac{p_i}{p_i + p_j}\Big)^r \Big(\frac{p_j}{p_i + p_j}\Big)^r\Big] - x\alpha\Big[p_j + \frac{2}{n(n-1)}\Big]. \end{aligned} \tag{3.10}$$

Denote the first part of the right-hand side of (3.10) − up to the right bracket − by A; we obtain for it a simple (and coarse) bound by letting the summation over m extend to infinity (all the terms are positive), and after some cancellations obtain

$$A \le \tfrac{1}{2}\Big(\frac{p_i - p_j}{p_i + p_j}\Big)^2 \sum_{r\ge 0} (2r+1)\binom{2r}{r} \Big(\frac{p_i}{p_i + p_j}\Big)^r \Big(\frac{p_j}{p_i + p_j}\Big)^r = \frac{p_i + p_j}{2(p_i - p_j)}. \tag{3.11}$$

Thus, getting back to $\Delta^{(m)}$, we only need the solution of the inequality

$$\sum_{i,j \in S} \Big\{ \frac{p_i + p_j}{2(p_i - p_j)} - x\alpha\Big[p_j + \frac{2}{n(n-1)}\Big]\Big\} \le 0, \tag{3.12}$$

for an x independent of i, j. Setting $m^* = \lfloor x \rfloor$, we get the value presented in equation (3.2). □

While Lemma 3.1 is by design dependent on the exact values of the *rpv* elements, we can easily obtain the following "distribution-free" bound:

Theorem 3.2: For an n-long linear list and an arbitrary *rpv*, the *CS* approaches the optimal cost to within a factor of $1 + \alpha$ after at most m^* accesses, where

$$m^* = \frac{n(n-1)(1+\alpha)^2}{16\alpha^2}\,. \tag{3.13}$$

Proof: All we need to do is reconsider equation (3.12) and find

$$\max_{i,j\in S} \frac{\frac{p_i+p_j}{2\,(p_i-p_j)}}{\alpha\left(p_j+\frac{2}{n(n-1)}\right)}\,.$$

Using the constraint given in (3.3), it is obvious that the solution x to the inequality $G(x;\,p_i,\,p_j)\le 0$, would obtain its maximal value for such p_i, p_j that satisfy $p_i - p_j = 2t$. Hence, we search for

$$\max_{0<p_j<1} \frac{\alpha(p_j+\frac{2}{n(n-1)})+p_j}{2\alpha^2(p_j+\frac{2}{n(n-1)})^2}\,. \tag{3.14}$$

The maximum is immediately obtained at $p_j = 2(1-\alpha)/[n(n-1)(1+\alpha)]$, and the value obtained upon substitution is the one given in equation (3.13). □

While Theorem 3.2 fills the claim we made, it was in an important sense disappointing: this is particularly evident in the way the bound was obtained: it was determined by a pair of p_i and p_j chosen for their closeness, and typically, at a very low value for $p_i + p_j$. However, our understanding of the synergy of the *CS* and the *irm* tells us that precisely for such pairs of records the correct order is unimportant. Therefore we should expect that the above bound will prove to be quite excessive. Still, in the only general approach that was open to us, such bounds are handed out.

An idea how unsatisfactory these bounds really are is furnished by a case so simple that inequality (3.1) can be solved directly for m^*:

Theorem 3.3: If the *rpv* is l-uniform, i.e

$$p_i = \begin{cases} 1/l & 1\le i\le l \\ 0 & l<i\le n \end{cases} \tag{3.16}$$

then inequality (3.1) is satisfied for all $\alpha > 0$ with

$$m^* = \log\frac{n-l}{\alpha(l+1)} \Big/ \log\frac{l}{l-1}\,. \tag{3.17}$$

For l in excess of approximately 20 an adequate approximation is

$$m^* = l\log\frac{n-l}{\alpha\, l}\,. \tag{3.18}$$

A "uniform" bound, that is useful when the value of l is not known *a priori*, and that holds for all $\alpha \ge 0.01$ is $m* = en$, $e = 2.7182818\ldots$.

Proof: For *rpv*'s in that category, equations (2.1–3) give

$$C_m(CS|\boldsymbol{p}) = 1 + \sum_{1\le i<j\le l}\sum_{k=0}^{m}\binom{m}{k}(1-\frac{2}{l})^{m-k}(\frac{2}{l})^k\cdot\frac{1}{l} + \sum_{i=1}^{l}\sum_{j=l+1}^{n}\frac{1}{2l}\,(1-\frac{1}{l})^m = \frac{l+1}{2} + \frac{(n-l)(1-\frac{1}{l})^m}{2}\,.$$

Thus,

$$\frac{C_m(CS|\boldsymbol{p})}{C(OPT|\boldsymbol{p})} = 1 + \frac{(n-l)(1-1/l)^m}{l+1}\,, \tag{3.19}$$

and equation (3.17) follows. Equation (3.18) is obtained by replacing the logarithm in the

denominator by its first-order approximation. For the "uniform" bound: Let $l = xn, 0 < x \le 1$, substitute $m = en$ and define

$$f(n) \equiv \frac{n(1-x)(1-\frac{1}{xn})^{en}}{xn+1} \quad .$$

For a fixed given x,

$$f(n) = \frac{n(1-x)(1-\frac{1}{xn})^{\frac{xen}{x}}}{xn+1} \le \frac{n(1-x)(e^{-e})^{1/x}}{xn+1} \equiv g(n) \; .$$

As $g(n)$ is monotonically increasing in n, we get

$$f(n) \le \lim_{n\to\infty} g(n) = \frac{(1-x)(e^{-e})^{1/x}}{x} \le \max_{0<x\le 1}\left(\frac{(1-x)(e^{-e})^{1/x}}{x}\right) = 0.008905 < 0.01 \; .$$

□

As we do not know how to demonstrate analytically such tight bounds for any other, less degenerate *rpv*'s, we tried a numerical example, to assess the quality of the above bounds.

We considered the Zipf distribution, where $p_i = 1/iH_n$. In this distribution the larger probabilities are relatively few and well differentiated, so one would expect the *CS* to be well suited to it. This was important, since computing $C_m(CS|\boldsymbol{p})$ from equations (2.1−3) is unpleasant at best. This is how part (a) of Table 1 was computed. Parts (b) and (c) of the table represent the

$n\backslash\,\alpha$	0.05	0.1	0.15	0.2
5	1.04	0.40	0.28	0.16
10	0.73	0.29	0.16	0.10
15	0.57	0.21	0.12	0.071
20	0.40	0.18	0.095	0.058
25	0.40	0.15	0.082	0.051

(a)

$n\backslash\,\alpha$	0.05	0.1	0.15	0.2
5	7.00	2.48	1.48	0.96
10	4.98	1.93	1.11	0.75
15	4.11	1.56	0.88	0.59
20	3.43	1.30	0.74	0.49
25	2.97	1.12	0.64	0.43

(b)

$n\backslash\,\alpha$	0.05	0.1	0.15	0.2
5	22.05	6.05	2.93	1.8
10	24.8	6.81	3.31	2.03
15	25.73	7.06	3.43	2.1
20	26.18	7.18	3.49	2.14
25	26.46	7.26	3.53	2.16

(c)

Table 1

The required number of reorganizations under *CS* to approach the optimum to within $1 + \alpha$. The values are given as multiples of n^2. Using Zipf distribution.

other two ways of obtaining m^*: The first was obtained assuming knowledge of the *rpv*, from equation (3.2). The second was obtained directly from the *distribution-free* bound of equation (3.13), and displays markedly different behavior.

4. The Limited-Counters Scheme

We present a "truncated version" of the original *CS*, and compare its performance with that of the MTF and the optimal ordering. Consider the "Limited-Counters Scheme" (LCS) in which $c_{\max}$ is the maximal value any of the counters is allowed to assume: A counter c_i which reaches $c_{\max}$ remains unchanged when further requests are made for R_i. From this follows that R_i does not change its position when further accessed. Thus, as m increases, a longer and longer prefix of the list stays "frozen".

Let $C_m(LCS|\boldsymbol{p}, c_{\max})$ denote the expected access-cost of the $m+1$st request under LCS. Its value is closely related to $C_m(CS|\boldsymbol{p})$, naturally:

Lemma 4.1: If the list is initially ordered arbitrarily, so that each of the arrangements is equi-probable and $c_{\max} = c$, then:

(i) for $1 \le m \le 2c$ $\quad C_m(LCS|\boldsymbol{p}, c) = C_m(CS|\boldsymbol{p})$. (4.1)

(ii) for $m > 2c$:

$$C_m(LCS|\boldsymbol{p}, c) = 1 + \sum_{i=1}^{n} p_i \sum_{\substack{j=1 \\ j\neq i}}^{n} \Big[\sum_{k=0}^{2c} \binom{m}{k} (1 - p_i - p_j)^{m-k} f_{ij}(k) + \sum_{k=2c+1}^{m} \binom{m}{k} (1 - p_i - p_j)^{m-k} (p_i + p_j)^{k-2c} f_{ij}(2c) \Big]. \tag{4.2}$$

where

$$f_{ij}(k) = \sum_{r=\lfloor k/2 \rfloor + 1}^{k} \binom{k}{r} p_j^r p_i^{k-r} + I_k \frac{1}{2} \binom{k}{k/2} (p_i p_j)^{k/2} . \tag{4.3}$$

Proof: (i) When $m \le 2c$, the LCS produces exactly the same permutations as the original *CS* rule does. Hence, the equality (4.1) follows.

(ii) For $m > 2c$ we follow the steps used in the proof of Lemma 2.1, except that once $2c$ references have been made to a pair, *at least* one of the counters must have reached c, and their relative position is thereby fixed. □

Now consider the limiting cost:

Lemma 4.2: The asymptotic expected access-cost under LCS is given by:

$$C(LCS|\boldsymbol{p}, c) \equiv \lim_{m\to\infty} C_m(LCS|\boldsymbol{p}, c) = 1 + \sum_{i=1}^{n} p_i \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{\sum_{r=c+1}^{2c} \binom{2c}{r} p_j^r p_i^{2c-r} + \frac{1}{2} \binom{2c}{r} (p_i p_j)^c}{(p_i + p_j)^{2c}} . \tag{4.5}$$

Proof: The asymptotic cost can be obtained by taking the implied limit in the cost function defined in Lemma 4.1. It is simpler to note that as the number of references increases, *every* pair reaches the limit of $2c$ references, that determine their relative order, and then the sum over j in equation (4.5) is merely the probability that R_j precedes R_i.

□

Clearly, one expects the performance of the *LCS* to improve monotonically as c increases, since we are using, in a consistent way, more information. A formal proof is easy to do: it only requires showing the monotonicity in c of the second line in the right-hand side of equation (4.2). This in turn follows by defining $a_{ij} \equiv p_i/p_j$; the above expression is homogeneous in a_{ij}, and the monotonicity can be shown on a power-by-power basis.

Certainly we shall have for all $c \geq 1$ and $m > 2c$

$$C_m(LCS|\boldsymbol{p}, c) \geq C_m(CS|\boldsymbol{p}). \tag{4.7}$$

The extent of the difference is not easy to characterize in any formal way, beyond the explicit expressions. Some understanding of its possible range will be provided by the bounds we give below. We bring now a result that suggests that the limited *CS* is of interest, even in its most stunted version.

Theorem 4.3: Let $C_m(MTF|\boldsymbol{p})$ denote the expected cost of the $m+1$st request under the MTF rule. Then for all $c \geq 1$, $m \geq 1$

$$C_m(LCS|\boldsymbol{p}, c) \leq C_m(MTF|\boldsymbol{p}). \tag{4.8}$$

□

The Theorem has a computational proof, which has some intrinsic interest. But there is a more appealing one: it starts with what is possibly the most surprising aspect of Theorem 4.3; namely, when $c = 1$, the relation (4.8) realizes the equality – that is, the LCS with $c = 1$ provides exactly the cost of the MTF scheme. This is shown by the following consideration: given that a pair of records R_i and R_j were at all accessed, the events {between the two, R_i was the *last* to be referenced} and {between the two, R_i was the *first* to be referenced} have equal probabilities, due to the stationarity of the *irm*. These events would have R_i precede R_j under the *MTF* and $LCS(c = 1)$ rules, respectively. Similar equal-treatment arguments take care of the cases with fewer references. The monotonicity of $C_m(LCS|\boldsymbol{p}, c)$ in c, mentioned above, completes the proof.

Using the LCS with $c = 1$ means a record will be moved at most once – when it is first accessed: it is then moved to the tail of the sublist of records that have been accessed before. The Theorem does not claim this is a particularly good scheme; Lemma 4.2 implies we could do better, and the following Theorem outlines by how much better.

Theorem 4.4: For any $n \geq 2$, an *rpv* $\boldsymbol{p}$ and $c \geq 2$ the asymptotic expected cost under the LCS is bound as follows:

$$\frac{C(LCS|\boldsymbol{p}, c)}{C(OPT|\boldsymbol{p})} \leq 1 + \max_{a \geq 1} \frac{(a-1)\left[\sum_{r=0}^{c} \binom{2c}{r} a^r - \tfrac{1}{2}\binom{2c}{c} a^c\right]}{(1+a)^{2c}}. \tag{4.13}$$

Proof: The proof consists in manipulating expressions we have obtained above. First, from equation (4.5) we find, by splitting the sum over the index j and using the shorthand notations $a_{ij} \equiv p_j/p_i$ and $g(c, a) \equiv \Sigma_{r=0}^{c} \binom{2c}{r} a^r$,

$$C(LCS|\boldsymbol{p}, c) = \sum_{i=1}^{n} \sum_{j=1}^{i} p_i \frac{(1 + a_{ij})^{2c} + (a_{ij} - 1)[g(c, a_{ij}) - \frac{1}{2}\binom{2c}{c} a_{ij}^c]}{(1 + a_{ij})^{2c}}. \tag{4.14}$$

Note that due to the convention of numbering the records in decreasing order of access probabilities, $a_{ij} \geq 1$, for all i and j. Using equation (2.13) and the relation (that holds for all positive numbers) $\Sigma_i a_i / \Sigma_j b_j \leq \max_k (a_k/b_k)$, we find

$$\frac{C(LCS|\boldsymbol{p},c)}{C(OPT|\boldsymbol{p})} \le \max_{1\le i\le n} \frac{1}{i} \sum_{j=1}^{i} \frac{(1+a_{ij})^{2c} + (a_{ij}-1)[g(c,a_{ij}) - \frac{1}{2}\binom{2c}{c}a_{ij}^{c}]}{(1+a_{ij})^{2c}},$$

and replacing the sum over j by i times its maximum, we get

$$\frac{C(LCS|\boldsymbol{p},c)}{C(OPT|\boldsymbol{p})} \le \max_{1\le i\le n} \max_{1\le j\le i} \frac{(1+a_{ij})^{2c} + (a_{ij}-1)[g(c,a_{ij}) - \frac{1}{2}\binom{2c}{c}a_{ij}^{c}]}{(1+a_{ij})^{2c}}. \tag{4.15}$$

Finally, replacing the double maximization by allowing any $a \ge 1$, relation (4.13) follows. □

Comment: Theorem 4.4 provides a bound that involves, for small values of c, low-degree polynomials. Thus, it is easy to show, by finding directly the maxima of the respective right-hand sides of equation (4.13), that the ratios $C(LCS|\boldsymbol{p},c)/C(OPT|\boldsymbol{p})$ for $c=2,3,4$, are bound by 1.3156, 1.2175 and 1.1733, respectively.

Furthermore, these bounds are valid for *all rpv*s. We expect this universality, as well as the cavalier manner in which the bound was proved to imply that for most distributions the results would even be better. This was tested out for two cases, the geometric and Zipf distributions – the asymptotic cost was computed from equation (4.5) and compared with $C(OPT|\boldsymbol{p})$ of equation (2.13). Tables 2 and 3 show the results. The most interesting feature is possibly the behavior of the ratio—or the excess cost as displayed in Table 2—with λ: $C(LCS|\boldsymbol{p},c)$ is very close to the optimal cost when the distribution is either *very* skew or nearly flat, and less so in intermediate cases. This fits perfectly with our understanding of the strengths of the *CS* approach.

$\lambda \backslash c$	2	3	4
0.1	0.019	0.005	0.002
0.3	0.08	0.045	0.027
0.5	0.125	0.080	0.058
0.7	0.158	0.055	0.079
0.9	0.180	0.132	0.104
0.99	0.038	0.037	0.036

Table 2
The excess cost $C(LCS|\boldsymbol{p},c)/C(OPT|\boldsymbol{p}) - 1$ for a list of 25 items
and geometrical access probabilities: $p_i = (1-\lambda)\lambda^{i-1}/(1-\lambda^n)$

$c \backslash n$	5	10	15	20	25
2	0.130	0.159	0.170	0.176	0.179
3	0.096	0.113	0.119	0.122	0.124
4	0.075	0.087	0.091	0.093	0.094
5	0.061	0.070	0.074	0.075	0.076
6	0.051	0.059	0.062	0.063	0.064
7	0.044	0.051	0.053	0.055	0.055
8	0.038	0.045	0.047	0.048	0.049
9	0.033	0.040	0.042	0.043	0.044
10	0.030	0.036	0.038	0.039	0.039

Table 3
The excess cost $C(LCS|\boldsymbol{p},c)/C(OPT|\boldsymbol{p}) - 1$ for lists of items
accessed with a Zipf *rpv*, given by $p_i = 1/iH_n$.

5. Discussion and Open Problems

We have argued the case for the Counter Scheme as the best policy for organizing a linear list when a stationary *irm* may be assumed to hold. Under such conditions it is the preferred method of list reorganization, with the further provision that there is adequate storage in the data structure to hold the counters. Further, we addressed the question of determining the size of the counter fields when the reference probability vector is not known *a priori*. We discussed two possible solutions. The first one, the LCS, may reduce significantly the space complexity while nearly maintaining the high performance level of the full scheme; the upper bound on its possible expected cost, compared with the *optimal* scheme is very promising indeed − see the numbers in the comment following (4.15). The second approach also sets an upper bound on the space required for each of the counters, when the original counter scheme is implemented, by fixing a finite point of time at which the list reorganization may be stopped. Remembering that the linear list is acceptable as a data structure for frequently referenced data only when n is rather small – 6? 20? 50? unlikely to be much larger – we realize that the required storage, under the improved schemes, is hardly a serious consideration.

One of the interesting aspects of the analysis was the light it shed on tacit assumptions. For example, why is the result of Theorem 4.3, specialized for $c = 1$ surprising – as it is, at first sight? The technical justification for it is supplied in the proof, and it is rather intuitive − except that it invokes stationarity. The reason then we find it surprising that the "move once" LCS (with $c = 1$) is as good as the ever-busy *MTF*, is that we tend to prize recent information, knowing that stationary processes are no more common in the real world than unicorns.

An issue we have hardly touched upon is the cost of the actual reorganization scheme one chooses. This seems acceptable if we limit the discussion to schemes that use roughly the same operations. The three methods we discussed fill this condition, except the time-limited *CS*. A specific reason to avoid the issue was that when the reorganization cost is included in the reference cost, comparing different schemes becomes technology-dependent.

Do we actually propose the *CS* to be used in real-life applications? The answer depends on the extent to which real life satisfies our model assumptions. The tritest of these is the independence of successive references. For various reasons, of which we believe the reader could supply several examples, our computational processes do not normally behave in this manner. The assumption may be reasonably if approximately true when the reference string is the result of superposing numerous sources – it often happens then that the serial correlation in the resulting sequence is negligible. When the independence premise fails, the best policy clearly depends on the failure mode: when the references exhibit a periodic nature, over the entire list, Move To the Rear (deterministically!) suggests itself; a strong locality property promotes MTF, with more complicated patterns producing ever more elaborate optimal permutation rules.

Next comes the assumption of fixed probabilities. Presumably, even when the the values wobble, but without changing their sorted order, the *CS* remains the method of choice. When this is not the case, the issue is open.

Indeed, the field swarms with open questions. To name a few: Incorporating external information about some of the probabilities (this is addressed to some extent in a forthcoming paper); Devising an efficient adaptive scheme for an early termination of the *CS*; Detection of a possible dependence structure in the reference model; For a given structure – obtaining the optimal strategy; To combine the last two: fashioning an adaptive control mechanism to fit a reorganization rule as knowledge of the reference model evolves; For a time-varying reference model – determining and detecting critical points. Under *any* model, how should dynamic lists, with insertions and deletions, be handled? With similar data structures, such as doubly-linked

lists, or circular lists, is the *CS*, suitably modified, still effective? At what stage do other data structures, more expensive to maintain but with shorter search sequences, become competitive? How are they to be reorganized? etc. etc.

For some of those, such as dynamic lists, even a proper definition of the problem remains elusive. Answers to any of these questions promise to be interesting both from the practical and theoretical points of view.

REFERENCES

Bellow M.E.: An investigation of self-organizing heuristics for doubly-linked lists. Proceedings of the 25th Allerton Conf. pp. 64−65, (1987)

Bentley J.L., McGeogh C.C.: Amortized analyses of self-organizing sequential search heuristics. *Commun. ACM 28*, 4, 1985, pp. 404−411.

Bitner J.R.: Heuristics that dynamically organize data structures. *SIAM J. Comput. 8*, 1, 1979, pp. 82−110.

Burville P.J., Kingman J.F.C.: On a model for storage and search. *J. Appl. Probab. 10*, 3, 1973, pp. 697−701.

Gibbons J.D., Olkin I., Sobel M.: Selecting the One Best Category for the Multinomial Distribution. In *Selecting and Ordering Populations: A New Statistical Methodology*, J. Wiley, 1977, pp. 158−186, 443−450.

Hendricks W.J.: An account of self-organizing systems. *SIAM J. Comput. 5*, 4, 1976, pp. 715−723

Hester J.H., Hirschberg D.S.: Self-organizing linear search. *ACM Comput. Surv. 17*, 3, 1985, pp. 295−312.

M. Hofri, H. Shachnai: On the Optimality of the Counter Scheme for Dynamic Linear Lists. *Inf. Proc. Lett.*, **37**, 175−179, (1991)

Knuth D.E.: A "self-organizing" file. In *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*, Addison-Wesley, Reading, MA. 1973, pp. 398−399.

Lam K., Siu M.K., Yu C.T.: A generalized counter scheme. *Theor. Comput. Sci. 16*, 3, 1981, pp. 271−278.

McCabe J.: On serial files with relocatable records. *Oper. Res. 13*, 1965, pp. 609−618

Oommen B.J., Hansen E.R.: List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations. *SIAM J. Comput. 16*, 4, 1987, pp. 705−716.

Oommen B.J., Hansen E.R., Munro J.I.: Deterministic Move-to-Rear list organizing strategies with optimal and expedient properties. In *Proceedings of the 25th Allerton Conf.* pp. 54−63, (1987).

Rivest R.: On self-organizing sequential search heuristics. *Commun. ACM 19*, 9, 1976, pp. 63−67.

Sleator D.D., Tarjan R.E.: Amortized efficiency of list update and paging rules. *Commun. ACM 28*, 2, 1985, pp. 202−208.

Topkis D.M.: Reordering Heuristics for Routing in Communication Networks. *J. Appl. Prob. 23*, 1986, pp. 130-143.