DETERMINISTIC OPTIMAL AND EXPEDIENT MOVE-TO-REAR LIST ORGANIZING STRATEGIES

by B.J. Oommen*, E.R. Hansen**
and J.I. Munro***

SCS-TR-124

October 1987

School of Computer Science
Carleton University
Ottawa, Ontario
CANADA K1S 5B6

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

* School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6
** Lockhead Missiles and Space Co. Inc., Sunnyvale, California 94086, USA
*** Dept. of Computer Science, University of Waterloo, Waterloo, Canada N2L 3G1

A preliminary version of this paper was presented at the Twenty-Fifth Allerton Conference in October 1987 at Urbana, Illinois.

DETERMINISTIC OPTIMAL AND EXPEDIENT MOVE-TO-REAR LIST ORGANIZING STRATEGIES†

B. John Oommen*, E. R. Hansen**, and J. I. Munro***

ABSTRACT

Let $\mathfrak{L} = \{R_1, R_2, \ldots, R_N\}$ be a list of elements in which R_i is accessed with an (unknown) probability s_i. To minimize the cost of accessing the elements, it is advantageous if the elements are sorted in the descending order of the access probabilities. Attempts to achieve this have been made in which a simple list reordering operation is performed on every access.

In this paper we present two simple self-organizing strategies. The strategies are **deterministic** and **absorbing** in their Markovian representation and are completely counter-intuitive, in that they are of a Move-To-Rear (MTR) flavour. Whereas the first of the schemes requires linear space (space proportional to the number of elements in the list), the second requires only constant space. We show that the former scheme is optimal, independent of the distribution of the access probabilities. By this we mean that although the list could converge to one of its N! configurations, by suitably performing the move-to-rear operation, the probability of converging to the right arrangement can be made as close to unity as desired. The second scheme requiring constant space is shown to be expedient. We conjecture its optimality.

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

---

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

* School of Computer Science, Carleton University, Ottawa : K1S 5B6, CANADA.

** Lockhead Missiles and Space Co. Inc., Sunnyvale, California : 94086, USA.

***Department of Computer Science, University of Waterloo, Waterloo N2L 3G1, CANADA.

I. INTRODUCTION

Suppose we are given a set of elements $\mathfrak{R} = \{R_1, \ldots, R_N\}$. 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.

This problem of having a file organize itself has been studied extensively. Various reviews on self-organizing strategies have been published over the years due to Gonnet et al [4], Rivest [13] and more recently by Hester and Hirschberg [7]. Since the literature in this field is so extensive, we refer the reader to these comprehensive surveys of the papers and results in the area [4, 7, 13]. Rather than describe the various strategies in any detail, we shall briefly highlight some of their main features.

McCabe [10, pp.398-399; 11] was the first to propose a solution to this problem. His solution rendered the list dynamically self-orginizing 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,

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

Many other researchers [3, 4-6, 14] have also extensively studied the MTF rule and various properties of its limiting convergence characteristics are available in the literature.

McCabe [10, pp.298-399; 11] 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, 10, 11, 13] but particular important is the work of Rivest [13] and Tenenbaum et al [1, 15] who extensively studied this rule and suggested its gereralizations - the Move-k-Ahead rule and the POS(k) rule. In the former, the accessed element is moved k-positions forward towards 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 $s_2 = s_3 = ... = s_N = (1-s_1) / (N-1)$. Their results seem to strengthen Rivest's conjecture.

Almost all of the schemes discussed in the literature are represented by Markov chains which are ergodic [1, 3, 4, 6, 8, 13, 15]. 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 **single** 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.

If the access probabilities $\{s_i\}$ are time invariant, clearly, the ergodic Markovian representation of a list organizing scheme is an undesirable property. All the efforts taken for an element to learn its right position can be easily annulled by the occurrence of events (such as the access of the least frequently accessed element) which have finite probability. Indeed, for a particular scheme, even if the probability of the list being in the unique optimal configuration is large, the probability of it remaining in that configuration for any length of time **can** be arbitrarily close to zero.

As opposed to ergodic representations, Markovian behaviour can also be absorbing [9]. In an absorbing Markov chain, the chain converges to one of a (finite) set of absorbing barriers. The problem of computing the probability of converging to any **particular** barrier involves evaluating what is called the first passage probability [11] and is well known in the theory of stochastic processes.

The literature reports of only two absorbing list organizing schemes [7, 12] both of which are due to Oommen and Hansen, and have been described in [12]. Both the algorithms were essentially of a learning flavour and were **stochastic**, and in them,

the elements of the list adaptively learned to find their place. The first algorithm in [12] is essentially a move-to--front algorithm with the exception that on the n^{th} access, the accessed element is moved to the front of the list with a probability f(n). This probability is systematically **decreased** every time an 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). This renders the Markovian representation of the procedure to be absorbing, as opposed to ergodic. The organization of the list gets "absorbed" into one of the N! orderings. It was shown in [12] that the scheme is expedient, i.e., if $s_i > s_j$, then the probability of absorption into an arrangement in which R_i precedes R_j is always greater than 0.5.

The second algorithm presented in [12] is far more powerful. In this case, the algorithm is a **move-to-rear** scheme in which the accessed element R_j is moved to the **rear** of the list with a probability q_j. This quantity q_j is progressively decremented every time **the element** is accessed. As before, ultimately there is no move operation performed on the list. Thus this scheme too is absorbing in its Markovian representation and so could converge into any one of its N! arrangements. However, in this case, it was shown that the probability of converging to the optimal arrangement can be made as close to **unity** as desired.

The latter technique requires more workspace than the traditionally used methods. However, apart from the latter algorithm being much more accurate than all the algorithms reported in the literature, it is also computationally more efficient. This is because the access of an element requires the update of **exactly** one probability (namely the probability q_j). Further, unlike the contemporary algorithms, since the list operations are essentially stochastic, a list operation is **not** 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**.

It must be noted, of course, that by virtue of the strong law of large numbers, an absorbing scheme would have resulted if frequency counters (which counted the number of accesses of the elements) were maintained, and the list maintained as a sorted one based on the access frequencies. However, the superiority of the latter scheme of [12] to that of using counters has been proven in [12], primarily in terms of the number of updates and in terms of the probability of asymptotically performing move operations.

Apart from the schemes in [12] being absorbing, the most interesting property of the optimal one is that it is a **move-to-rear** scheme. This is indeed quite counter-intuitive that by stochastically moving an accessed element to the **rear** of the list an optimal list arrangement could result. But the fact is that such a stochastic scheme is obtainable.

In this paper, we shall present two new absorbing list organizing schemes. The schemes are very simple to implement. They are deterministic and again of a **move-to-rear** flavour. Additionally, the number of move operations performed in both the schemes are **exactly** equal to the number of elements in the list, primarily because a move operation is performed on an element **exactly** once. Whereas the first of the schemes requires linear space (space proportional to the number of elements in the list), the second requires only constant space (i.e., space independent of the number of elements in the list). The most powerful property of the former scheme is that it is optimal, in that the probability of being absorbed into the optimal configuration can be made as close to unity as desired. The second scheme requiring constant space is shown to be expedient. In other words we shall show that if $s_i > s_j$, then the probability of being absorbed into an arrangement in which R_i precedes R_j is greater than 0.5. We conjecture its optimality.

Although both of the schemes described in this paper are conceptually Move-to-Rear schemes, the potential drawbacks that a MTR scheme could possess are absent in both of them primarily because they can be implemented very conveniently by introducing **a single** additional pointer. In a pure MTR scheme the elements which are accessed very infrequently would linger at the front of the list till they are themselves to the rear of the list. These infrequently accessed elements could thus potentially prove to be a drag on the entire access strategy. However, this can be easily overcome by maintaining two list pointers, the first pointer always pointing to the front of the original list, and the second pointer pointing to an interior element of the list. This interior element is indeed the element which was **first** moved to the rear. The details of this implementation and the access strategy will be discussed later.

As in the literature concerning the theory of adaptive learning, we shall use the terms algorithm, rule and scheme interchangeably. Further, for the sake of simplicity we shall assume that the access probabilities of the records are distinct. The cases when they are nondistinct are those when there are **many** optimal configurations. A remark about these cases will be made appropriately in the body of the paper.

II. A DETERMINISTIC LINEAR SPACE MOVE-TO-REAR STRATEGY

We shall now present a deterministic absorbing list organizing strategy and show that it is asymptotically optimal. The idea behind the scheme is extremely straightforward.

Let T be any fixed integer. Associated with every element R_i is an integer memory location x_i which is initially set to zero. Whenever the element R_i is accessed x_i is incremented, but **no** list reorganization is performed. Whenever the value of x_i is exactly T, the element R_i is moved to the **rear** of the list. Subsequently, the element R_i is neither moved nor its associated location x_i updated. This describes the list organization strategy completely.

For the sake of completeness, we shall algorithmically describe the above process.

ALGORITHM Deterministic-MTR-LSpace

Input : A sequence of accesses on the list $\mathfrak{L} = \{R_1, R_2, \ldots, R_N\}$. R_i is accessed with an unknown access probability s_i. This is read in using procedure ReadInput.

Output : A reorganized list.

Memory Requirements :

(i) An integer memory location x_i associated with every R_i.

(ii) A boolean variable M_i associated with R_i. M_i indicates if R_i has already been moved.

(iii) An integer constant T.

Method :

```
For i = 1 to N do
        x_i = 0
        M_i = False
EndFor
Repeat
        ReadInput (R_i)
        If Not (M_i) then
                x_i = x_i + 1
                If (x_i = T) then
                        Move-to-Rear (R_i)
                        M_i = True
                EndIf
        EndIf
Forever
```

End Algorithm Deterministic-MTR-LSpace

We shall now prove the properties of the above strategy.

Theorem I

The deterministic Move-to-Rear Scheme described above is absorbing. Furthermore, the total number of list reorganizing operations done is exactly N.

Proof :

The second assertion is obvious. The absorbing nature of the Markovian representation of the chain follows from the fact that after every element has been accessed T times the list remains in its final configuration, and no more list reorganizing is performed. • • •

The power of the above scheme lies primarily in the fact that it is not just the **expected** number of move operations which is finite. This was, of course, true of the stochastic schemes described in [12]. In this case however, the number of moves performed, **not being a random variable**, is exactly equal to the size of the list.

The following results concerning the probability of being absorbed in the optimal configuration is more powerful. We shall show that the scheme is asymptotically optimal, i.e., the probability of being absorbed in the optimal configuration can be made as close to unity as desired.

Theorem II

For any distinct indices u, v ε {1, 2, ..., N}, let $\xi_{u,v}$ be the event that either R_u or R_v is accessed. Further, let ${}_uP_v$ be the **conditional** probability that R_u ultimately precedes R_v given $\xi_{u,v}$. Then,

(i) ${}_uP_v$ can be made as to close to unity as desired if $s_u > s_v$ and,

(ii) ${}_uP_v$ is exactly 0.5 if $s_u = s_v$.

Proof :

Let $p = s_u / (s_u + s_v)$, and $q = s_v / (s_u + s_v)$. Clearly, p is the (conditional) probability of accessing R_u given $\xi_{u,v}$. Similarly, q is the (conditional) probability of accessing R_v given $\xi_{u,v}$, and hence p+q=1.

For the rest of this proof we shall assume that all the probabilities and expressions are conditional probabilities, conditioned on $\xi_{u,v}$.

Let $\langle x_u(n), x_v(n)\rangle$ refer to the number of times R_u and R_v are accessed up to and including the n^{th} time instant respectively conditioned on $\xi_{u,v}$. For the sake of simplicity of notation, in the latter pair, we shall not specify the time instant 'n', and hence, unless explicitly stated, the pair $\langle x_u, x_v\rangle$ will refer to the pair $\langle x_u(n), x_v(n)\rangle$.

Clearly, at the n^{th} time instant if $\langle x_u, x_v\rangle$ equals $\langle T, j\rangle$ for $j < T$ then, since R_u has been accessed T times, it will be moved to the rear of the list, and subsequently the element R_u will **precede** R_v in the **ultimate** order of the elements. Conversely, if $\langle x_u, x_v\rangle = \langle i, T\rangle$ for $i < T$, then, **ultimately**, R_u will succeed R_v in the list. Thus, the count on the number of accesses on R_u and R_v conditioned on $\xi_{u,v}$ obeys the following Markov chain $\mathfrak{M} = (\Phi, F)$, where,

(i) Φ is the set of states, and every state is a pair $\langle i, j \rangle$, where $\langle i, j \rangle \in \{0, 1, \ldots, T\} \times \{0, 1, \ldots, T\} - \{\langle T, T \rangle\}$, and

(ii) F is the transition function specifying the probability of moving from one state to another. The transition function F can be seen to have the following form :

$$F_{\langle i, j \rangle, \langle k, m \rangle} = \begin{cases} p & \text{if } k = i+1 \leq T;\ m = j < T \\ q & \text{if } k = i < T;\ m = j+1 \leq T \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

Furthermore,

$$F_{\langle i, j \rangle, \langle i, j \rangle} = 1 \quad \text{if exactly one of i,j is T} \qquad (2)$$

The transition map of the Markov chain is given schematically in Figure I.

The Markov chain is clearly absorbing with the set of absorbing barriers being the set $\{\langle T, j \rangle \mid j < T\} \cup \{\langle i, T \rangle \mid i < T\}$. The probability of converging to a configuration in which R_u precedes R_v is exactly equal to the probability of being absorbed into any of the states in the set $\{\langle T, j \rangle \mid j < T\}$.

Using the well known theory of absorbing Markov chains [9] one can compute the first passage probabilities for converging in the various absorbing states. In this case the expressions can be derived rather easily, since the transition map of the internal (non-absorbing) states is cycle-free. Thus, consider the conditional probability of converging in $\langle T, j \rangle$ given $\xi_{i,j}$. The Markov chain converges in $\langle T, j \rangle$ if out of the first (T+j-1) accesses, T-1 of them are for R_u and j of them for R_v. The last access has to be for R_u if it is to converge in $\langle T, j \rangle$. Since $\xi_{i,j}$ is the conditional event given, the distribution is binomial, and hence,

$$\Pr[\text{Convering in } \langle T, j \rangle \mid \xi_{i,j}] = \binom{T+j-1}{j} p^T q^j$$

Hence, the conditional probability of R_u ultimately preceding R_v (conditioned on $\xi_{i,j}$) is ${}_uP_v$, where,

$$ {}_uP_v = \sum_{j=0}^{T-1} \binom{T+j-1}{j} p^T q^j = p^T \sum_{j=0}^{T-1} \binom{T-1+j}{j} q^j $$

Observe that ${}_uP_V$ is the sum of the probabilities of being absorbing in <T, j> where j<T.

We shall now consider the limiting value of ${}_uP_V$. Indeed, by virtue of Lemma I, ${}_uP_V$ can be made as close to unity as desired by correspondingly increasing T, since,

$$ \lim_{T\to\infty} {}_uP_v = 1 $$

Lemma I

$$ \text{Let } f_n(p) = p^n \sum_{k=0}^{n-1} \binom{n-1+k}{k} q^k $$

where $p+q = 1$ and $0 \le p \le 1$. Then,

$$ \lim_{n\to\infty} f_n(p) = 1 \quad \text{for } 1/2 < p \le 1 $$

$$ = 0 \quad \text{for } 0 \le p < 1/2 $$

$$ = 1/2 \quad \text{for } p = 1/2. $$

Proof :

The proof is rather involved and is included in the Appendix. • • •

The final theorem regarding the asymptotic optimality of the scheme now follows directly.

Theorem III

Let $\mathfrak{L} = \{R_1, R_2, \ldots, R_N\}$ be the list of elements with distinct access probabilities $\{s_1, \ldots, s_N\}$. Then, the probability of the list converging into the optimal configuration can be made as close to unity as desired.

Proof :

For every distinct pair u, v $\{1,\ldots, N\} \times \{1,\ldots, N\}$, we have shown that ${}_uP_V$ can be made as close to unity as desired if $s_u > s_v$. Let P^* be defined as below :

P^* = Pr [list converging to optimal configuration]

Then, since the records are independently drawn according to the distribution $\{s_i\}$,

$$P^* = \prod_{u \neq v} \Pr[R_u \text{ precedes } R_v \mid s_u > s_v] = \prod_{s_u > s_v} {}_uP_v$$

The result that P^* tends to unity follows since each ${}_uP_v$ does so as the variable $T \rightarrow \infty$.

• • •

It is interesting to note that if the access probabilities are not distinct, then there is no **unique** optimal configuration. In this case, by virtue of Theorem II, the scheme will converge to **any one** of the optimal configurations, and the probability measure of doing this is equally divided between the individual optimal configurations. This is intuitively appealing.

III. A DETERMINISTIC CONSTANT SPACE MOVE-TO-REAR STRATEGY

We now present a deterministic absorbing list organizing strategy that requires only **constant** space and show that it is expedient. In other words we shall show that for this scheme, if $s_i > s_j$, then the probability of being absorbed into an arrangement in which R_i precedes R_j is greater than 0.5. We conjecture that the optimality of this scheme.

Let T be any fixed integer and Z be a memory location having two integer fields, z_1 and Z_2. Whenever the element R_i is accessed Z_1 is set to i. Furthermore, if Z_1 was previously set to i, Z_2 is incremented. Otherwise the value of Z_2 is reset to zero. Whenever the value of Z_2 is exactly T, the element R_i (i.e., the element whose index is stored in Z_1) is moved to the **rear** of the list. Subsequently, the element R_i is neither moved nor its associated location x_i updated. This describes the list organization strategy completely. Observe that essentially all that this scheme does is that it moves a record R_i to the rear of the list after it has been accessed T **consecutive** times.

For the sake of completeness, we shall algorithmically describe the above process.

ALGORITHM Deterministic-MTR-CSpace

Input : A sequence of accesses on the list $\mathfrak{L} = \{R_1, R_2, \ldots, R_N\}$. R_i is accessed with an unknown access probability s_i. This is read in using procedure ReadInput.

Output : A reorganized list.

Memory Requirements :

(i) An memory location Z with two integer fields Z_1 and Z_2. Z_1 stores the index of the last accessed record, and Z_2 counts the number of times the record whose index is Z_1 has been consecutively accessed.

(ii) A boolean variable M_i associated with R_i. M_i indicates if a MTR (i.e., a list operation) has been performed on R_i. Although this implies that the algorithm requires linear space (since a memory location M_i has to be maintained for ever record R_i), we shall later show that by the introduction of a **single** extra pointer the implementation can be rendered to require **only** constant space. However, the pseudo-code is more understandable if we assume that every record has a single additional **bit** of information storing whether it has been moved or not.

(iii) An integer constant T.

Method :

```
For i = 1 to N do
        Mi = False
EndFor
Z1 = - ∞
Repeat
        ReadInput (Ri)
        If Not (Mi) then
                If (Z1 = i ) then
                        Z2 = Z2 + 1
                Else
                        Z2 = 1
                        Z1 = i
                Endif
                If (Z2 = T) then
                        Move-to-Rear (Ri)
                        Mi = True
                Endif
        Endif
Forever
End Algorithm Deterministic-MTR-CSpace
```

We shall now prove the properties of the above strategy.

Theorem IV

The constant space deterministic MTR strategy described by Algorithm Deterministic-MTR-CSpace above is expedient.

Proof :

It is required to prove that if $s_U > s_V$, then the probability of absorption into an arrangement in which R_i precedes R_j is always greater than 0.5.

For any distinct indices u, v ε {1, 2, ..., N}, let $\xi_{U,V}$ be the event that either R_U or R_V is accessed. Further, let ${}_UP_V$ be the **conditional** probability that R_U ultimately precedes R_V given $\xi_{U,V}$. Then, we shall show that by choosing the value of T to be its minimum value of unity,

(i) ${}_UP_V$ is strictly greater than 0.5 if $s_U > s_V$ and,

(ii) ${}_UP_V$ is exactly 0.5 if $s_U = s_V$.

Let $p = s_U / (s_U + s_V)$, and $q = s_V / (s_U + s_V)$. Clearly, p is the (conditional) probability of accessing R_U given $\xi_{U,V}$. Similarly, q is the (conditional) probability of accessing R_V given $\xi_{U,V}$, and hence p+q=1. Since T is set to **unity**, R_U ultimately precedes R_V if it is accessed **before** R_V, since then it will be the first of the two which is first moved to the rear of the list . Subsequently, of course, R_V will be moved to the rear of the list and will thus ultimately **succeed** R_V in the final ordering of the elements. Thus, the probability of R_U ultimately preceding R_V takes the value of unity whenever R_U is accessed before R_V, and it takes the value of zero whenever R_V is accessed before R_U. Explicitly, since the accesses are independent,

$$\Pr[\, R_U \text{ ultimately precedes } R_V \,] = \begin{cases} 1 & \text{w.p. } s_U \text{ if } \xi_{U,V} \text{ is true.} \\ 0 & \text{w.p. } s_V \text{ if } \xi_{U,V} \text{ is true.} \end{cases}$$

Since the accesses of R_U and R_V partition the conditional space specified by $\xi_{U,V}$, the quantity ${}_UP_V$ which is the conditional probability of R_U ultimately precedeing R_V conditioned on $\xi_{U,V}$ being true is obtained by the laws of elementary probability to be $p = s_U/(s_U + s_V)$. The theorem follows since p is strictly is strictly greater than 0.5 if $s_U > s_V$ and is exactly 0.5 if $s_U = s_V$. • • •

Apart from the above scheme being expedient we also believe the following conjecture.

Conjecture I

The constant space deterministic MTR strategy described by Algorithm Deterministic-MTR-CSpace above is asymptotically optimal.

For any distinct indices u, v ε {1, 2, ..., N}, let $\xi_{u,v}$ be the event that either R_u or R_v is accessed. Further, let ${}_uP_v$ be the **conditional** probability that R_u ultimately precedes R_v given $\xi_{u,v}$. Then, the proof of the conjecture will involve proving that if T is increased indefinitely,

(i) ${}_uP_v$ is can be made as close to unity as desired if $s_u > s_v$, and,

(ii) ${}_uP_v$ is exactly 0.5 if $s_u = s_v$.

The proof of this conjecture is by no means trivial. It involves the solution of a random walk, in which the random "walker" can make single step traversals of length up to and including T/2. We hope to prove this conjecture in the near future.

Should this conjecture be true, the constant space deterministic MTR strategy will be the **ideal** list organizing strategy. The strategy requires only a constant amount of extra memory, performs exactly one list reordering operation per record and converges in an **absorbing** (as opposed to ergodic) mode to a final ordering. Furthermore, should this conjecture be true, the probability that this final ordering is the **optimal** ordering can be made as close to unity as desired.

IV. IMPLEMENTATION DETAILS

It is interesting to understand how the MTR schemes discussed in this paper may be actually implemented. To explain the implementation details, we shall merely specify these details for the case of the linear space scheme described in Section II. The principles are analogous for the constant space scheme described in Section III.

Consider the linear space scheme described in Section II. Observe that initially the schemes **actually** moves the most frequently accessed elements to the rear of the list. Thus, the elements which are accessed very infrequently would linger at the front of the list till they are accessed T times, and thus prove to be a drag on the access strategy. However, this can be easily overcome by maintaining two list pointers FrontOld and FrontNew. The first pointer, FrontOld always points to the front of the original list, and the second pointer, FrontNew, points to the element in the list which was **first** moved to the rear.

When an access on R_i is made, the list pointed to by FrontNew is first searched, and if the element is found, **neither** a counter update **nor** a list reorganization operation is executed. However, if the element R_i is not found in the list pointed to by FrontNew, a search is made by traversing the list which is pointed at by FrontOld. The latter traversal terminates at FrontNew and **may** result in a list reorganization.

The implementation of the scheme is shown in Figure II for a list of five elements, namely the list $\mathfrak{L} = \{R_1, R_2, ..., R_5\}$ in which $s_1 > s_2 > s_3 > s_4 > s_5$. In Figure II, we have considered a case when the variables $\{ x_i \}$ attain the value T in the order (1,2,3,4,5). Initially, on R_1 being accessed T times, it is moved to the rear of the list. Subsequently, on R_2 being accessed T times, **it** is moved to the rear of the list. As time proceeds, whenever $x_3 = T$, R_3 is moved to the rear of the list. However, to catalyze the accessing process, the list pointers FrontOld and FrontNew are maintained as described earlier. Note that ultimately, FrontOld and FrontNew point to the reorganized list.

As stated earlier, the same principles can be utilized for the implementation of the constant space scheme described in Section III.

IV. CONCLUSIONS

We have considered a list of elements $\mathfrak{L} = \{R_1, ..., R_N\}$ in which the element R_i is accessed with an unknown probability s_i. Two deterministic list organizing schemes have been proposed. The schemes have an absorbing Markovian representation, and he number of move operations performed in both the schemes is **exactly** equal to the number of elements in the list. Whereas the first of the schemes requires linear space (space proportional to the number of elements in the list), the second requires only constant space. We have shown that the former scheme is optimal. In other words, the probability of being absorbed into the optimal configuration can be made as close to unity as desired. The second scheme requiring constant space is shown to be expedient. We conjecture its optimality.

ACKNOWLEDGEMENTS

The authors are very grateful to David Ng who helped in preparing the final manuscript and the drawings.

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

[5] Hansen, E.R., A Table of Series and Products, Prentice Hall, Englewood Cliffs, N.J., 1975.

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

[7] Hester, J.H. and Hirschberg, D.S., "Self-Organizing Linear Search", ACM Computing Surveys, Vol. 17, 1985, pp.295-311.

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

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

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

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

[12] Oommen, B.J. and Hansen, E.R., "List Organizing Strategies Using Stochastic Move-to-front and Stochastic Move-to-Rear Operations", to appear in SIAM Journal on Computing. Preliminary results appear in 1986 International Conference on Database Theory, in Rome, Sept. 1986.

[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.

[16] Abramowitz, M. and Stegun I.A., Handbook of Mathematical Functions, National Bureau of Standards, Dover Publications, Inc., New York, 1970.

APPENDIX

Proof of Lemma I

Denote

$$f_n(p) = p^n \sum_{k=0}^{n-1} \binom{n-1+k}{k} q^k$$

where $p+q = 1$ and $0 \le p \le 1$. We intend to find

$$L(p) = \lim_{n \to \infty} f_n(p)$$

Solution. From equation (10.4.5) of [5],

$$f_n(p) = I_p(n, n)$$

where,

$$I_p(a, b) = \frac{B_p(a, b)}{B(a, b)}$$

and

$$B_p(a, b) = \int_0^p t^{a-1} (1-t)^{b-1} dt$$

is the incomplete beta function and

$$B(a, b) = \frac{\Gamma(a) \cdot \Gamma(b)}{\Gamma(a + b)}$$

is the (complete) beta function.

From the theory of Beta functions (see equations 26.5.10, 26.5.15, and 26.5.16 of [16]), we easily find that

$$f_{n+1}(p) - f_n(p) = \frac{(2n)!}{2 (n!)^2} p^n q^n (2p-1)$$

Summing this result,

$$\sum_{n=1}^{J} [f_{n+1}(p) - f_n(p)] = \frac{1}{2} (2p-1) \sum_{n=1}^{J} \frac{(2n)!}{(n!)^2} p^n q^n \qquad (A.1)$$

for arbitrary $J \ge 1$.

Because of cancellations in the sum in the left member, we have,

$$f_{J+1}(p) - f_1(p) = \frac{1}{2}(2p-1)\sum_{n=1}^{J}\frac{(2n)!}{(n!)^2}p^n q^n$$

now $f_1(p) = p$ and hence,

$$L(p) = \lim_{J\to\infty} f_{J+1}(p) = p + \frac{1}{2}(2p-1)\sum_{n=1}^{\infty}\frac{(2n)!}{(n!)^2}p^n q^n \qquad \text{(A.2)}$$

From equation (5.24.31) of [5],

$$\sum_{n=1}^{\infty}\frac{(2n)!}{(n!)^2}p^n q^n = (1-4pq)^{-1/2} - 1$$

$$= \begin{cases} \dfrac{1}{1-2p} - 1 & \text{for } 0 \le p < 1/2 \\ \dfrac{1}{2p-1} - 1 & \text{for } 1/2 < p \le 1 \end{cases} \qquad \text{(A.3)}$$

Therefore, (A.2) -(A.4) yield,

$$L(p) = \begin{cases} 0 & \text{for } 0 \le p < 1/2 \\ 1 & \text{for } 1/2 < p \le 1 \end{cases} \qquad \text{(A.4)}$$

We now consider the case $p = 1/2$. Note that

$$B(n,n) = \int_0^1 t^{n-1}(1-t)^{n-1}\,dt$$

Dividing the interval of integration into two halves,

$$B(n,n) = \int_0^{1/2} t^{n-1}(1-t)^{n-1}\,dt + \int_{1/2}^{1} t^{n-1}(1-t)^{n-1}\,dt$$

Replacing t by 1-t in the second integral we find it equals the first integral and hence

$$B(n, n) = 2 \int_0^{1/2} t^{n-1} (1 - t)^{n-1} \, dt$$

$$= 2 B_{1/2}(n, n).$$

Therefore, for all positive n,

$$f_n(1/2) = I_{1/2}(n, n) = 1/2$$

Thus,

$$L(1/2) = \lim_{n \to \infty} f_n(1/2) = 1/2.$$

Combining the above results,

$$\begin{aligned} L(p) &= 0 && \text{for } 0 \le p < 1/2 \\ &= 1/2 && \text{for } p = 1/2 \\ &= 1 && \text{for } 1/2 < p \le 1. \end{aligned}$$

• • •

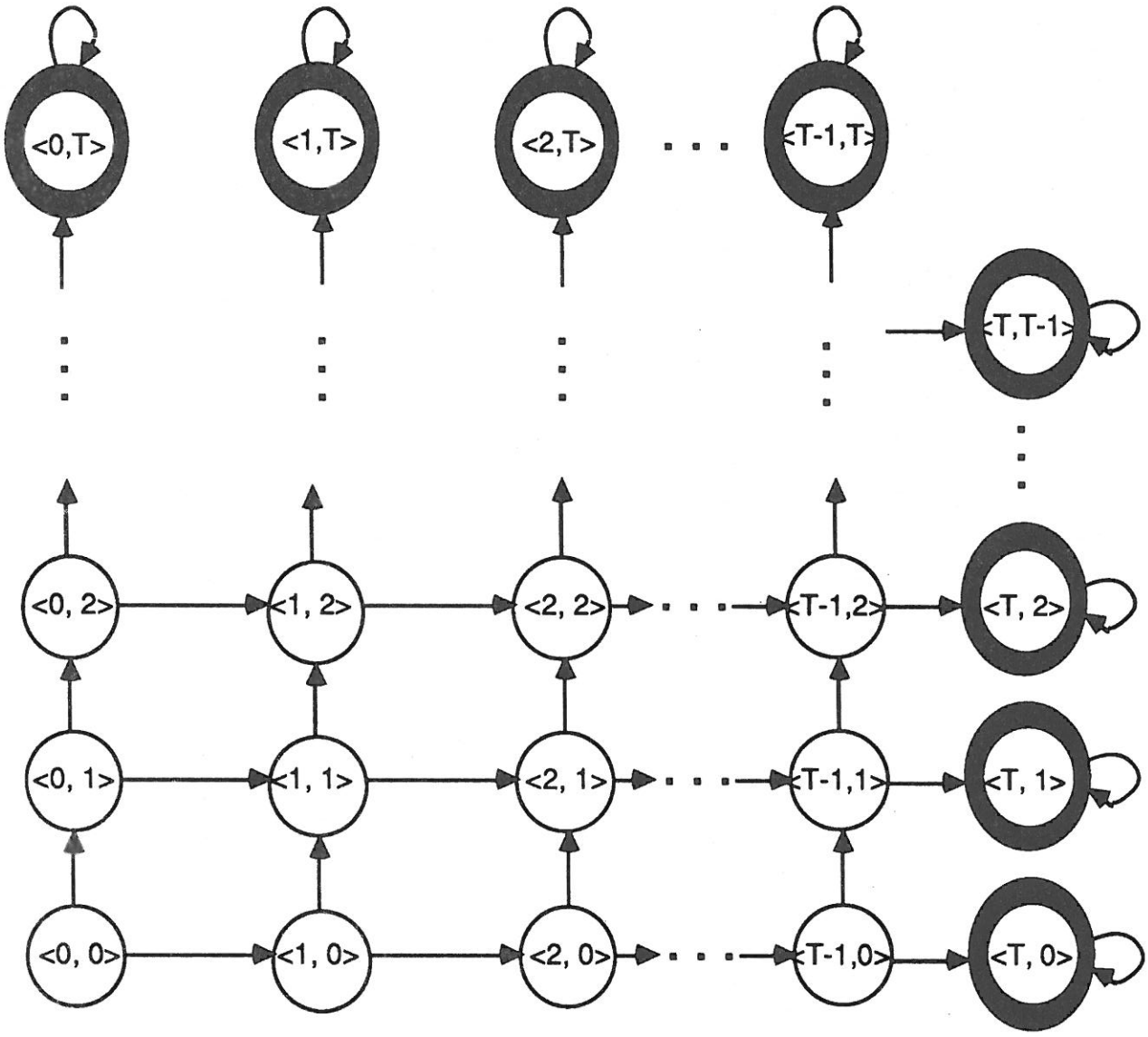

Figure I : The transition map of the Markov Chain defined in Theorem II. All the horizontal transitions are with probability p, and the vertical transitions are with probability q, where, p+q=1. The self-loops are with probability one. The bold circles represent the absorbing states.

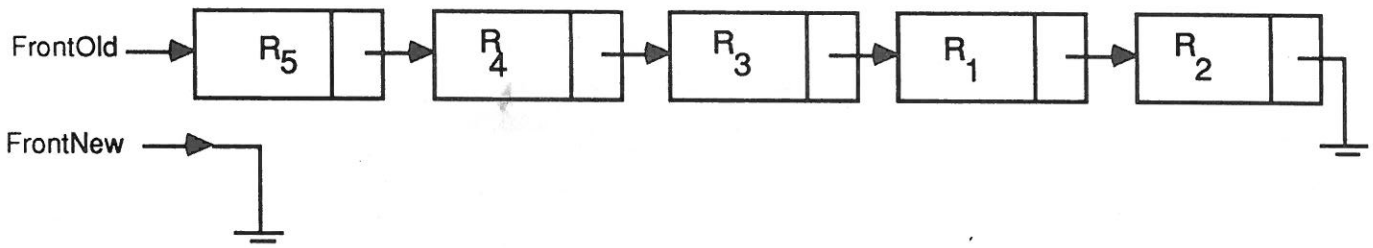

Figure II (a)

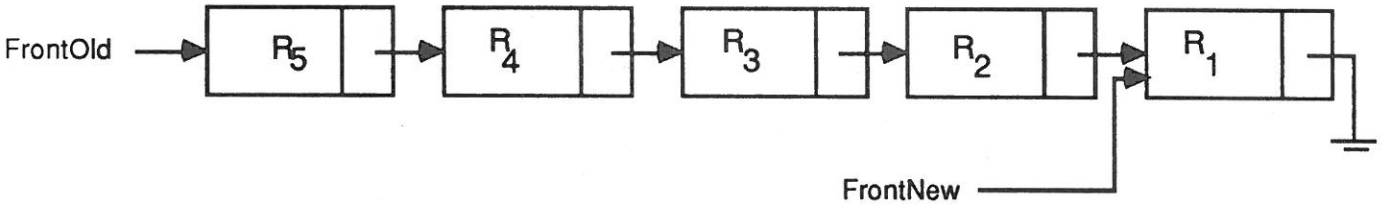

Figure II (b)

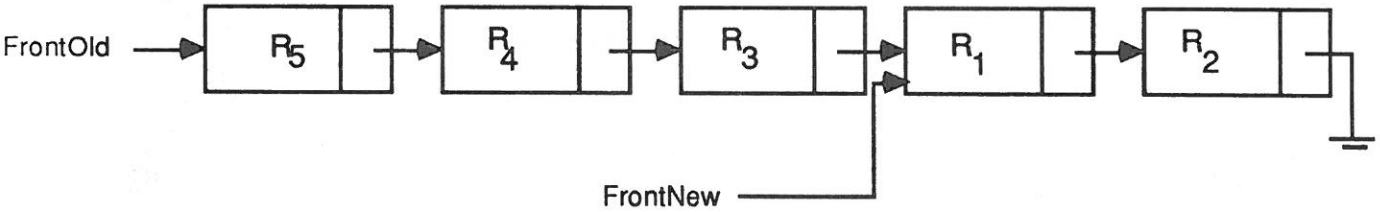

Figure II (c)

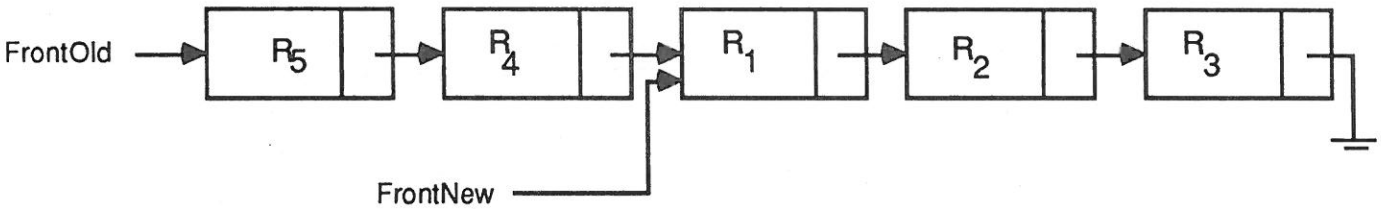

Figure II (d)

Figure II : Implementation of the Move-to-Rear Scheme. FrontOld points to the original Head of the list and FrontNew points to the element which was first moved to the rear. The list is traversed from FrontNew, and if the element sought for is not obtained, it is then traversed from FrontOld.

Carleton University, School of Computer Science
Bibliography of Technical Reports
Publications List (1985 -->)

School of Computer Science
Carleton University
Ottawa, Ontario, Canada
KIS 5B6

SCS-TR-66 **On the Futility of Arbitrarily Increasing Memory Capabilities of Stochastic Learning Automata**
_____ B.J. Oommen, October 1984. Revised May 1985.

SCS-TR-67 **Heaps in Heaps**
_____ T. Strothotte, J.-R. Sack, November 1984. Revised April 1985.

SCS-TR-68 **Partial Orders and Comparison Problems**
out-of-print M.D. Atkinson, November 1984. See Congressus Numerantium 47 ('86), 77-88

SCS-TR-69 **On the Expected Communication Complexity of Distributed Selection**
_____ N. Santoro, J.B. Sidney, S.J. Sidney, February 1985.

SCS-TR-70 **Features of Fifth Generation Languages: A Panoramic View**
_____ Wilf R. LaLonde, John R. Pugh, March 1985.

SCS-TR-71 **Actra: The Design of an Industrial Fifth Generation Smalltalk System**
_____ David A. Thomas, Wilf R. LaLonde, April 1985.

SCS-TR-72 **Minmaxheaps, Orderstatisticstrees and their Application to the Coursemarks Problem**
_____ M.D. Atkinson, J.-R. Sack, N. Santoro, T. Strothotte, March 1985.

SCS-TR-73 **Designing Communities of Data Types**
Wilf R. LaLonde, May 1985.
Replaced by SCS-TR-108

SCS-TR-74 **Absorbing and Ergodic Discretized Two Action Learning Automata**
out-of-print B. John Oommen, May 1985. See IEEE Trans. on Systems, Man and Cybernetics, March/April 1986, pp. 282-293.

SCS-TR-75 **Optimal Parallel Merging Without Memory Conflicts**
_____ Selim Akl and Nicola Santoro, May 1985

SCS-TR-76 **List Organizing Strategies Using Stochastic Move-to-Front and Stochastic Move-to-Rear Operations**
_____ B. John Oommen, May 1985.

SCS-TR-77 **Linearizing the Directory Growth in Order Preserving Extendible Hashing**
_____ E.J. Otoo, July 1985.

SCS-TR-78 **Improving Semijoin Evaluation in Distributed Query Processing**
_____ E.J. Otoo, N. Santoro, D. Rotem, July 1985.

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-79
_____
On the Problem of Translating an Elliptic Object Through a Workspace of Elliptic Obstacles
B.J. Oommen, I. Reichstein, July 1985.

SCS-TR-80
_____
Smalltalk - Discovering the System
W. LaLonde, J. Pugh, D. Thomas, October 1985.

SCS-TR-81
_____
A Learning Automation Solution to the Stochastic Minimum Spanning Circle Problem
B.J. Oommen, October 1985.

SCS-TR-82
_____
Separability of Sets of Polygons
Frank Dehne, Jörg-R. Sack, October 1985.

SCS-TR-83
out-of-print
Extensions of Partial Orders of Bounded Width
M.D. Atkinson and H.W. Chang, November 1985. See Congressus Numerantium, Vol. 52 (May 1986), pp. 21-35.

SCS-TR-84
_____
Deterministic Learning Automata Solutions to the Object Partitioning Problem
B. John Oommen, D.C.Y. Ma, November 1985

SCS-TR-85
out-of-print
Selecting Subsets of the Correct Density
M.D. Atkinson, December 1985. To appear in Congressus Numerantium, Proceedings of the 1986 South-Eastern conference on Graph theory, combinatorics and Computing.

SCS-TR-86
_____
Robot Navigation in Unknown Terrains Using Learned Visiblity Graphs. Part I: The Disjoint Convex Obstacles Case
B. J. Oommen, S.S. Iyengar, S.V.N. Rao, R.L. Kashyap, February 1986

SCS-TR-87
_____
Breaking Symmetry in Synchronous Networks
Greg N. Frederickson, Nicola Santoro, April 1986

SCS-TR-88
_____
Data Structures and Data Types: An Object-Oriented Approach
John R. Pugh, Wilf R. LaLonde and David A. Thomas, April 1986

SCS-TR-89
_____
Ergodic Learning Automata Capable of Incorporating Apriori Information
B. J. Oommen, May 1986

SCS-TR-90
_____
Iterative Decomposition of Digital Systems and its Applications
Vaclav Dvorak, May 1986.

SCS-TR-91
_____
Actors in a Smalltalk Multiprocessor: A Case for Limited Parallelism
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-92
_____
ACTRA - A Multitasking/Multiprocessing Smalltalk
David A. Thomas, Wilf R. LaLonde, and John R. Pugh, May 1986

SCS-TR-93
_____
Why Exemplars are Better Than Classes
Wilf R. LaLonde, May 1986

SCS-TR-94
_____
An Exemplar Based Smalltalk
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-95
_____
Recognition of Noisy Subsequences Using Constrained Edit Distances
B. John Oommen, June 1986

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-96 **Guessing Games and Distributed Computations in Synchronous Networks**
J. van Leeuwen, N. Santoro, J. Urrutia and S. Zaks, June 1986.

SCS-TR-97 **Bit vs. Time Tradeoffs for Distributed Elections in Synchronous Rings**
M. Overmars and N. Santoro, June 1986.

SCS-TR-98 **Reduction Techniques for Distributed Selection**
N. Santoro and E. Suen, June 1986.

SCS-TR-99 **A Note on Lower Bounds for Min-Max Heaps**
A. Hasham and J.-R. Sack, June 1986.

SCS-TR-100 **Sums of Lexicographically Ordered Sets**
M.D. Atkinson, A. Negro, and N. Santoro, May 1987.

SCS-TR-102 **Computing on a Systolic Screen: Hulls, Contours, and Applications**
F. Dehne, J.-R. Sack and N. Santoro, October 1986.

SCS-TR-103 **Stochastic Automata Solutions to the Object Partitioning Problem**
B.J. Oommen and D.C.Y. Ma, November 1986.

SCS-TR-104 **Parallel Computational Geometry and Clustering Methods**
F. Dehne, December 1986.

SCS-TR-105 **On Adding *Constraint Accumulation* to Prolog**
Wilf R. LaLonde, January 1987.

SCS-TR-107 **On the Problem of Multiple Mobile Robots Cluttering a Workspace**
B. J. Oommen and I. Reichstein, January 1987.

SCS-TR-108 **Designing Families of Data Types Using Exemplars**
Wilf R. LaLonde, February 1987.

SCS-TR-109 **From Rings to Complete Graphs - Θ(n logn) to Θ(n) Distributed Leader Election**
Hagit Attiya, Nicola Santoro and Shmuel Zaks, March 1987.

SCS-TR-110 **A Transputer Based Adaptable Pipeline**
Anirban Basu, March 1987.

SCS-TR-111 **Impact of Prediction Accuracy on the Performance of a Pipeline Computer**
Anirban Basu, March 1987.

SCS-TR-112 **ε-Optimal Discretized Linear Reward-Penalty Learning Automata**
B.J. Oommen and J.P.R. Christensen, May 1987.

SCS-TR-113 **Angle Orders, Regular n-gon Orders and the Crossing Number of a Partial Order**
N. Santoro and J. Urrutia, June 1987.

SCS-TR-115 **Time Is Not a Healer: Impossibility of Distributed Agreement in Synchronous Systems with Random Omissions**
N. Santoro, June 1987.

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-116 **A Practical Algorithm for Boolean Matrix Multiplication**
M.D. Atkinson and N. Santoro, June 1987.

SCS-TR-117 **Recognizing Polygons, or How to Spy**
James A. Dean, Andrzej Lingas and Jörg-R. Sack, August 1987.

SCS-TR-118 **Stochastic Rendezvous Network Performance - Fast, First-Order Approximations**
J.E. Neilson, C.M. Woodside, J.W. Miernik, D.C. Petriu, August 1987.

SCS-TR-120 **Searching on Alphanumeric Keys Using Local Balanced Trie Hashing**
E.J. Otoo, August 1987.

SCS-TR-121 **An O(√n) Algorithm for the ECDF Searching Problem for Arbitrary Dimensions on a Mesh-of-Processors**
Frank Dehne and Ivan Stojmenovic, October 1987.

SCS-TR-122 **An Optimal Algorithm for Computing the Voronoi Diagram on a Cone**
Frank Dehne and Rolf Klein, November 1987.

SCS-TR-123 **Solving Visibility and Separability Problems on a Mesh-of-Processors**
Frank Dehne, November 1987.

SCS-TR-124 **Deterministic Optimal and Expedient Move-to-Rear List Organizing Strategies**
B.J. Oommen, E.R. Hansen and J.I. Munro, October 1987.

SCS-TR-125 **Trajectory Planning of Robot Manipulators in Noisy Workspaces Using Stochastic Automata**
B.J. Oommen, S. Sitharam Iyengar and Nicte Andrade, October 1987.

SCS-TR-126 **Adaptive Structuring of Binary Search Trees Using Conditional Rotations**
R.P. Cheetham, B.J. Oommen and D.T.H. Ng, October 1987.