ON DOUBLY LINKED LIST REORGANIZING HEURISTICS*

D.T. H. Ng and B.J. Oommen

SCS-TR-151
February, 1989

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

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

ON DOUBLY LINKED LIST REORGANIZING HEURISTICS†

D.T.H. Ng and B.J.Oommen
School of Computer Science,
Carleton University
Ottawa, Ont. K1S 5B6
CANADA.

ABSTRACT

The class of memoryless heuristics for maintaining a doubly linked list in an approximately optimal order is studied. Initially the problem, its representation and the constraints on doubly linked lists are defined. Various mappings and theorems that relate singly linked list and doubly linked list heuristics are presented, and a new heuristic referred to as the Swap heuristic for the doubly linked list is introduced. The Swap heuristic is shown to be more efficient than the Move-To-End heuristic.

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

I. INTRODUCTION

In the development of self-organizing list structures, a problem that has been extensively studied concerns that of a singly linked (or sequential) list. The literature on adaptive sequential list organization is indeed extensive; a detailed survey on this topic can be found in [Hest85]. In particular, the Move-To-Front and Transposition heuristics have been intensively analysed by Hendricks [Hend72], Bitner [Bitn79], Knuth [Knut73] and Rivest [Rive76], to name a few. Also, heuristics which use additional amount of memory apart from the memory used by the list structure itself were studied by many scientists and numerous papers were published concerning this topic. We refer the reader to the following papers : McCabe [McCa65], Kan *et. al.* [Kan80], Gonnett *et. al.* [Gonn79], Oommen *et. al.* [Oomm87] and Oommen *et. al.* [Oomm88].

The Move-To-Front heuristic (MTF) operates on the principle that when the accessed record is found, it is moved to the front of the list. On the other hand, the transposition heuristic exchanges the accessed record with the immediately preceding record; nothing is done if the accessed record is at the front of the list. Since these two schemes have contributed to the foundation of the study of self-organizing structures, we will use these as bench-marks to verify some of the theorems in this paper.

Although the theory of self-organizing singly linked lists is well developed, very little work has been done to study the adaptive restructuring of doubly linked lists. Currently, to our knowledge, the only paper found in this area is by Matthews *et. al.* [Matt80] which gives some preliminary but powerful results in this area.

The problem concerning the doubly linked list can be defined as follows : consider a set of N records $R_1, R_2, \ldots R_N$ which we specify in an arbitrary order π, so that R_i is in position $\pi(i)$ counting from the left to the right for $1 \leq i \leq N$. A search for R_i can begin at either end of the list, examining each position in turn until R_i is found. At each instant of time, a key K_i is presented at one end of the list and the system is asked to retrieve the associated record R_i. The goal of the heuristic is to asymptotically minimize the average (or expected) number of comparisons required in the data retrieval process.

We assume that the record R_i is requested at the left end and the right end of the list with a probability $s_{i,L}$ and $s_{i,R}$ respectively. Each access is stochastically independent and the access probabilities of the records are time invariant, with the constraint that,

$$\sum_{i=1}^{N} s_{i,L} + \sum_{i=1}^{N} s_{i,R} = 1, \quad (1)$$

where, $s_{i,L}$ = P(R_i is accessed from the left), and, (2)

$s_{i,R}$ = P(R_i is accessed form the right). (3)

If the key is presented from the left, the number of comparisons made is $\pi(i)$. Similarly, if the key is presented from the right, the number of comparisons is $N-\pi(i)+1$.

The heuristics which will be considered in this paper fall within the family of memoryless schemes. In other words, no extra memory variables are used beside the pointers used to maintain the list itself. Thus, any heuristic for the doubly linked list can be defined as a set of permutations $\tau = \{\tau_{i,L}, \tau_{i,R}\}$ such that the permutation $\tau_{i,L}$ is applied to the list when the element at the i^{th} position is accessed from the left, and the permutation $\tau_{i,R}$ is applied to the list if the element at the i^{th} position is accessed from the right.

Applications of doubly linked lists are indeed extensive due to the simplicity of implementing it compared to that of implementing other more complex data structures, and also due to the flexibility in accessing elements compared to the singly linked list. Many applications of doubly linked list are presented by Tremblay and Sorenson [Trem76].

In this paper, the underlying environment for any doubly linked structure is first analysed. It is shown that the same environment $\mathfrak{E}$ can be described using two representations and the essential equivalence of these representations is proved. Subsequently, we present two strategies by which an algorithm which is applicable for a singly linked list can be extended for the case of a doubly linked list. These two strategies involve, what we call, undirected and directed mappings respectively. When an undirected mapping is used, the doubly linked list heuristic executes the operation suggested by the singly linked list heuristic except that the information about the direction of access is not utilized. As opposed to this, if a directed mapping is used, the corresponding doubly linked list heuristic utilizes both the information concerning the accessed record and the end from which the record is accessed. Using these mappings the results of Matthews *et. al.* [Matt80] are first generalized. Subsequently, using a directed mapping, a new heuristic called the Swap heuristic for the doubly linked list is introduced and it is compared with the Move-To-End heuristic. The Swap heuristic is shown to be the more efficient of the two over **all** probability distributions. The theoretical results developed are strengthened by numerous simulation results presented at the end of the paper.

II. THE STUDY OF THE ENVIRONMENT

In order to make a thorough study of the adaptive doubly linked list problem, it is important to set up a comprehensive model for the underlying user environment $\mathfrak{E}$. The model which we consider here in this paper is one in which the event that occurs at one time instant is independent of the event occurring at any other instant. Each event has a probability of occurrence which is time invariant.

Every environment $\mathfrak{E}$ is fully defined by a set of possible events, and these constitute the accessing of the individual records from either of the two ends. To analyse the behaviour of an algorithm in any particular environment, it is imperative that the environment is modeled correctly and completely. By this we mean that for any model of the environment, it should be possible to assign a probability measure to every observable event; the only constraint to this assignment should be the laws of probability. We shall now present two such representations, the first will be called the total probability representation and the second, the conditional probability representation. Both these representations shall be proved to be equivalent, thus we can interchangeably use them since they are "isomorphic".

The representation referred to above, termed as the total probability representation is indeed the one defined by equation (1)-(3). The representation used by Matthews *et. al.* [Matt80] is the conditional probability representation.

We define a probability representation to be **stochastially complete** under the set of time invariant and independent environments, if for any environment in this set, there exists a set of probabilities such that every observable event has a consistent probability measure assigned to it.

The advantage of proving the two models to be stochastically complete is that whereas certain theorems can be proved using the total probability representation, other theorems can be more elegantly proved using the conditional probability representation. Thus, by making a one-to-one correspondence between the probabilities and the constraints imposed by both the representations, the analysis presented in the subsequent sections is rendered more easy.

Stochastic completeness is actually useful not only in the study of doubly linked lists. Indeed, it extends to the entire spectrum of problems involving probabilistic modelling. Not only does it provide a tool to simplify complex proofs by the availability of interchangeable representations, but it also gives the researcher a guide to compare the degree of freedom provided by different models and their representations.

Lemma I :

For a doubly linked list environment, the total probability representation given by (1)-(3) is stochastically complete.

Proof : This is true by definition since every single mutually exclusive event is assigned a **separate** probability mass, and the constraints together with the laws of elementary probabilities ensure the consistency of the assignments. •••

In the paper by Matthews *et. al.*[Matt80], the authors defined the following as a model for the underlying environment. Let,

p_i = probability of accessing R_i; (4)

$p_{i,L}$ = probability of accessing R_i from left **given** that R_i is accessed; (5)

$p_{i,R}$ = probability of accessing R_i from right **given** that R_i is accessed; (6)

$p_{i,L} + p_{i,R} = 1$; (7)

$$\sum_{i=1}^{N} p_i = 1. \quad (8)$$

Matthews *et. al.* [Matt80] used (4)-(8) to define the environment $\mathfrak{E}$. However, since the individual elementary probabilities defining an environment are those defined by (1)-(3), it is not entirely obvious that (4)-(8) completely represents **every** independent, time invariant distribution for $\mathfrak{E}$. This proof is included below. This is done not only for the sake of completeness but also to permit us to use both representations interchangeably.

Theorem I :

For a doubly linked list environment, the conditional probability representation given by (4)-(8) is stochastically complete.

Proof : This statement does not follow directly as in the case of Lemma I, but it remains to be proved that the equations (4)-(8) and the laws of probability guarantee stochastic completeness. However, since any environment can be modeled in a stochastically complete fashion by (1)-(3), we only need to show that the set of quantities defined by (4)-(6) can independently represent all the probabilities defined in (1)-(3) with the additional constraints given by (7) and (8).

The first part of the proof involves proving that (4)-(6) completely define all the probabilities given by (1)-(3). This is done by specifying the probability transitions between the two representations. Using the laws of probability, by explicitly writing the

probability associated with the various events and performing some rather straightforward algebraic manipulations, it can be shown that :

$$s_{i,L} = p_i \cdot p_{i,L}, \text{ and,} \tag{9}$$

$$s_{i,R} = p_i \cdot p_{i,R}. \tag{10}$$

Conversely $\{p_i\}$, $\{p_{i,L}\}$ can be obtained in terms of $\{s_{i,L}, s_{i,R}\}$ using similar computations as :

$$p_i = s_{i,L} + s_{i,R}, \tag{11}$$

$$p_{i,L} = \frac{s_{i,L}}{s_{i,L} + s_{i,R}}, \tag{12}$$

$$p_{i,R} = \frac{s_{i,R}}{s_{i,L} + s_{i,R}}. \tag{13}$$

Thus, (4)-(6) represent all the probabilities defined in (1)-(3).

We now show that no additional constraints exist in (4)-(8) by examining the constraints (7) and (8) in turn. Consider (7), which simplifies as :

$$p_{i,L} + p_{i,R} = \frac{s_{i,L}}{s_{i,L} + s_{i,R}} + \frac{s_{i,R}}{s_{i,L} + s_{i,R}}$$

which is unity, implying that the set $\{p_{i,L}, p_{i,R}\}$ are consistent.

Also, using the following two way implications, we assert :

$$\sum_{i=1}^{N} p_i = 1,$$

$$\Leftrightarrow \quad \sum_{i=1}^{N} p_i \, (p_{i,L} + p_{i,R}) = 1,$$

$$\Leftrightarrow \quad \sum_{i=1}^{N} p_i . p_{i,L} + \sum_{j=1}^{N} p_j p_{j,R} = 1,$$

$$\Leftrightarrow \quad \sum_{i=1}^{N} s_{i,L} + \sum_{j=1}^{N} s_{j,R} = 1.$$

Thus, the constraints (7) and (8) in the latter representation suffice for the model to be stochastically complete and the theorem is proved. •••

The above theorem not only provides a foundation for the model in any environment; it also provides us with the flexibility to translate between the two representations readily. The understanding of these representations make the theorems in the next section much more comprehensible.

III. DOUBLY LINKED LIST HEURISTICS AND THEIR ANALYSIS

Before trying to find a good heuristic, it is necessary to define a performance measure with which we can evaluate the efficiency of a heuristic. Let the **cost** measurement of a heuristic τ be $C_\tau(\mathfrak{E})$, the expected number of comparisons required for an access if the underlying user's environment is $\mathfrak{E}$. Notice that $\mathfrak{E}$ can be described in either of the representations mentioned in the previous section since they are both equivalent.

A heuristic τ is said to be a **more efficient** heuristic than σ if $C_\tau(\mathfrak{E}) \leq C_\sigma(\mathfrak{E})$ for any set of probability distributions $\mathfrak{E}$. Note that there exist heuristics for which $C_\tau(\mathfrak{E}) < C_\sigma(\mathfrak{E})$ for some $\mathfrak{E}$, and $C_\tau(\mathfrak{E}) \geq C_\sigma(\mathfrak{E})$ for another, and so this efficiency operator (i.e. if we view the term "more efficient" as an operator itself) is not functional over all pairs of heuristics. A heuristic is said to be optimal if it is more efficient than all other heuristics.

If the set of probability distributions $\mathfrak{E}$ is known *a priori*, we can arrange the records in such a way that the cost is minimized. Observe that this is trivial because it is easily seen that the cost of a permutation π is

$$\text{cost}(\pi) = \sum_{i=1}^{N} [s_{i,L}.\pi(i) + s_{i,R}.(N-\pi(i)+1)]$$

$$= \sum_{i=1}^{N} [(s_{i,L}-s_{i,R}).\pi(i) + s_{i,R}.(N+1)].$$

Since the second term of the summation is independent of $\pi(i)$, the **optimal order** is the permutation which arranges the list in the decreasing order of $(s_{i,L}-s_{i,R})$.

The cost measurement and the concept of having a 'more efficient' heuristic permits us to comparatively study various doubly linked list heuristics.

As a first step into studying doubly linked list heuristics, it is natural to look back at some singly linked list heuristics and try to extend some of these well known concepts to doubly linked lists. An obvious way to create a new doubly linked list heuristic is to use a transformation from a singly linked list heuristic into a corresponding doubly linked list heuristic. This approach is not new in the literature - it is indirectly used in [Matt80]. Our contribution in this context is the explicit classification of this transformation in terms of two types of mappings, namely, undirected and directed mappings.

Mappings are, in general, transformations by which a singly linked list heuristic can be changed into a doubly linked list heuristic. In an undirected mapping , the operations done on a doubly linked list involves only the accessed element but does not involve the dirction from which the element is accessed. As opposed to this, if a directed mapping is used, the reorganization is achieved using the information concerning the accessed element and the direction of access. An example of this will definitely help to clarify the point.

Consider the Move-To-Front heuristic for singly linked list. This heuristic leads to the Move-To-Left rule if an undirected mapping is used to get a doubly linked list scheme. This scheme is quite simply stated as follows. Whenever an element R_i is accessed, it is moved to the left end of the doubly linked list independent of the end from which the element is accessed. Indeed, it is not obvious whether such a rule has any advantages at all.

As opposed to the Move-To-Left rule, a directed mapping transforms the singly linked list Move-To-Front heuristic into a doubly linked list scheme which is called the Move-To-End heuristic. In this case the accessed element R_i is moved to the left end of the list if it is accessed from the left, and move to the right end of the list if it is accessed from the right. This is indeed the scheme presented in [Matt80].

In this section, we shall prove the advantages of various directed and undirected mappings.

III.1 Undirected Mappings of Singly Linked List Heuristics

Let $\tau^U = \{\tau_i\}$ be a singly linked list heuristic. This mean that if the record in position i is accessed, the data is reorganized according to the rule τ_i. Similarly, let $\tau^D = \{\tau_{i,L}, \tau_{i,R}\}$ be a doubly linked list heuristic. We say that an **undirected mapping** exists between τ^U and τ^D if $\tau_{i,L} = \tau_i = \tau_{i,R}$.

Should such a mapping exist, $E_{\tau D}[\pi(i)]$, the expected location of record R_i (counting from the left), no longer depends on the conditional probabilities $p_{i,L}$ and $p_{i,R}$, since the permutation performed will be the same regardless of the search direction. In other words, for a given environment $\mathcal{E}$, $E_{\tau U}[\pi(i)] = E_{\tau D}[\pi(i)]$.

Using the definition of an undirected mapping, we can compactly express one of the lemmas introduced by Matthews *et. al.* [Matt80]. The authors in [Matt80] proved that if an optimal algorithm τ^* exists for singly linked list and if $p_{i,L} = p > 1/2$, then the undirected mapping of τ^* is **also** an optimal heuristic for doubly linked list. But since Anderson [Ande85] proved the non-existence of such an optimal algorithm, clearly, the proof of the result of Mathews *et. al.* [Matt80] was implicitly rendered void. However, this does not completely render the paper [Matt80] ineffective. Indeed, the results of [Matt80] can be extended to consider the relative efficiency of families of heuristics used for doubly linked lists. To demonstrate this we prove the following result.

Theorem II :

If $p_{i,L}=p>1/2$ for i=1..N and if a singly linked list heuristic, τ^U, is a more efficient heuristic than ρ^U, then the corresponding doubly linked list heuristic, τ^D, under the undirected mapping is also more efficient than ρ^D.

Proof : For any doubly linked list heuristic σ with $p_{i,L}=p$ for i=1..N, the expected search time for any query is,

$$C_\sigma(\mathfrak{E}) = \sum_{i=1}^{N} p_i p E_\sigma[\pi(i)] + \sum_{i=1}^{N} p_i p (N - E_\sigma[\pi(i)] + 1)$$

$$= (2p-1) \sum_{i=1}^{N} p_i . E_\sigma[\pi(i)] + K, \qquad (14)$$

where K = (N+1) - (N+1)p.

From the search cost of singly linked lists, we know that

$$C_{\tau U}(\mathfrak{E}) = 1 + \sum_{i=1}^{N} p_i . E_{\tau U}[\pi(i)]$$

$$< 1 + \sum_{i=1}^{N} p_i . E_{\rho U}[\pi(i)]$$

$$= C_{\rho U}(\mathfrak{E}).$$

This implies that,

$$\sum_{i=1}^{N} p_i.E_{\tau U}[\pi(i)] < \sum_{i=1}^{N} p_i.E_{\rho U}[\pi(i)]. \qquad (15)$$

Using the property of undirected mapping, (15) gives

$$\sum_{i=1}^{N} p_i.E_{\tau D}[\pi(i)] < \sum_{i=1}^{N} p_i.E_{\rho D}[\pi(i)].$$

Thus,

$$C_{\tau D}(\mathbb{E}) = (2p-1) \sum_{i=1}^{N} p_i.E_{\tau D}[\pi(i)] + K$$

$$< (2p-1) \sum_{i=1}^{N} p_i.E_{\rho D}[\pi(i)] + K$$

$$= C_{\rho D}(\mathbb{E}).$$

The above inequality holds since $(2p-1) > 0$ for all $p > 0.5$, and the theorem is proved.

•••

A word about the power of Theorem II is not out of place, Matthews *et. al.* [Matt80] proved that if an **optimal** singly linked list heuristic existed, this could be transformed to yield an **optimal** doubly linked heuristic for the case when $p_{i,L} = p > 1/2$ for all i. By this theorem, we have now been able to present a hierarchy on the set of doubly linked list heuristics which are obtainable using undirected mappings. Theorem II ensures us that for any environment, if τ^U is superior to ρ^U as a singly linked list heuristic, then the corresponding heuristics (τ^D and ρ^D respectively) obtained using an undirected mapping will satisfy the property that τ^D is superior to ρ^D whenever $p_{i,L} = p > 1/2$. Although the result of Matthews *et. al.* is only true for the special case if an optimal algorithm existed, it is interesting to note the result presented above is an abstract generalization of their result and is valid for **all** pairs of heuristics.

Rivest [Rive76] proved that the Transposition heuristic is more efficient than the MTF heuristic. Using this fact in conjunction with Theorem II, we have the following corollary:

Corollary II.1

For all doubly linked lists, the Transposition heuristic under the undirected mapping is more efficient than the MTF heuristic under the undirected mapping.

Proof. The corollary follows directly from Theorem II using Rivest's result. •••

III.2. Directed Mapping of Singly Linked List Heuristics

The undirected mapping is not a very realistic generator to transform a singly linked list heuristic into a doubly linked list heuristic. This is because the permutation is not sensitive to the direction of access, which, intuitively, contains a lot of information and is thus an important component which can be used to improve the performance of the reorganization process.

Although directed mappings were informally described in the preamble of this section, we formally describe it as follows. A singly linked heuristic $\tau^U=\{\tau_i\}$ is defined to possess a **directed mapping** on a doubly linked list heuristic $\tau^D=\{\tau_{i,L}, \tau_{i,R}\}$, if $\tau_{i,L} = \tau_i = \tau_{N-i+1,R}$ for i=1..N.

Using the directed mapping, we can easily express the Move-To-End heuristic introduced by Matthews *et. al.*[Matt80] by transforming the Move-To-Front heuristic using the directed mapping into the Move-To-End heuristic. Using the Move-To-End heuristic, whenever a record R_i is found in position $\pi(i)$, if the search originates from the left end of the list, the list is rearranged by moving R_i to this end. The records in positions 1,..., $\pi(i)$-1 are each moved **down** one position. Otherwise, if the search originates from the right end of the list, the list is rearranged by moving R_i to the right end and the records in positions $\pi(i)+1$,..., N are each moved **up** one position.

Matthews *et. al.* [Matt80] proved the following results concerning the MTE heuristic. Let b(i, j) denote the asymptotic probability that R_i is to the left of R_j in the Move-To-End heuristic. This probability can be expressed as :

$$b(i, j) = \frac{s_{i,L} + s_{i,R}}{s_{i,L} + s_{i,R} + s_{j,L} + s_{j,R}}. \qquad (16)$$

Furthermore, the search cost of the MTE heuristic was shown to be :

$$C_{MTE}(\mathbb{E}) = 1 + 2\sum_{j=1}^{N}\left\{ s_{j,L}\sum_{i<j}\frac{s_{j,R} + s_{i,L}}{s_{i,L} + s_{i,R} + s_{j,L} + s_{j,R}} + s_{j,R}\sum_{i>j}\frac{s_{i,R} + s_{j,L}}{s_{i,L} + s_{i,R} + s_{j,L} + s_{j,R}}\right\} \qquad (17)$$

They also proved that the Move-To-End heuristic never requires more than twice the search time associated with the optimal ordering which is intuitively appealing considering the known results for singly linked lists.

Another well studied heuristic for sequential lists is the Transposition rule. We now extend this rule using a directed mapping to obtain a new heuristic for the doubly linked list. This heuristic will be called the **Swap heuristic**. With this scheme, whenever a record R_i is found in position $\pi(i)$, if the search originates from the left of the list, the list is rearranged by swapping the positions of records R_i and R_{i-1}, except when R_i is at the left end of the list, in which case, no records are moved. Otherwise, if the search originates from the right of the list, the list is rearranged by swapping R_i and R_{i+1}, except when R_i is at the right end of the list, in which case again, no records are moved.

The analysis of the Swap heuristic, which is new to the literature, is based on the finite time reversible Markov chain where each state is one of the N! possible orderings of the set of records $\{R_1,R_2,...,R_N\}$, and in which each ordering $R_{i_1}R_{i_2} \ldots R_{i_N}$ has a stationary probability $\text{prob}(R_{i_1}R_{i_2} \ldots R_{i_N})$ which can be expressed as:

$$\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_N}) = (s_{i_1,L} + s_{i_N,R})\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_N}) + \sum_{j=1}^{N-1}(s_{i_j,L} + s_{i_{j+1},R})\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_{j+1}}R_{i_j} \ldots R_{i_N}). \qquad (18)$$

In this light, we now present some of the properties of the Swap heuristic.

Theorem III :

Under the Swap heuristic, the stationary probabilities obey,

$$\frac{\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_j}R_{i_{j+1}} \ldots R_{i_N})}{\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_{j+1}}R_{i_j} \ldots R_{i_N})} = \frac{s_{i_j,L} + s_{i_{j+1},R}}{s_{i_{j+1},L} + s_{i_j,R}} \qquad (19)$$

Proof : If a set of state probabilities obeys (19), then it is a stationary distribution. By considering the state transitions of the underlying Markov chain described by the Swap heuristic, we shall show that (18) is an identity. Indeed, substituting (19) in (18) we have,

$$\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_N}) = (s_{i_1,L} + s_{i_N,R})\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_N}) + \sum_{j=1}^{N-1}(s_{i_{j+1},L} + s_{i_j,R})\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_N})$$

$$= \text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_N}).\sum_{j=1}^{N}(s_{i_j,L} + s_{i_j,R})$$

$$= \text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_N}).$$

Hence the theorem. ...

A similar result exists for the Transposition rule for the singly linked list (see [Rive76]). However, Theorem III is a far more general result. Indeed, in the special case when the probability of access from the right is zero (that is $s_{i,R}=0$ for all i), the Swap heuristic behaves identically to the Transposition heuristic and (19) becomes:

$$\frac{\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_j}R_{i_{j+1}} \ldots R_{i_N})}{\text{Prob}(R_{i_1}R_{i_2} \ldots R_{i_{j+1}}R_{i_j} \ldots R_{i_N})} = \frac{s_{i_j,L}}{s_{i_{j+1},L}}$$

which is the well acclaimed result due to Rivest.

Rivest [Rive76] also proved that the Transposition heuristic is always more efficient than the Move-To-Front heuristic. We can now derive the analogous result concerning the Swap heuristic.

Theorem IV :

The Swap heuristic is always more efficient than the Move-To-End heuristic.

Proof : Let $b_\tau(i, j)$ be the probability that R_i is to the left of R_j under the doubly linked list heuristic τ. The average cost can be expressed as

$$C_\tau(\mathbb{E}) = \sum_{j=1}^{N} s_{j,L} \left\{1 + \sum_{i \neq j} b_\tau(i,j)\right\} + \sum_{j=1}^{N} s_{j,R} \left\{N - \sum_{i \neq j} b_\tau(i,j)\right\}$$

$$= \sum_{j=1}^{N} \left\{N s_{j,R} + s_{j,L} + (s_{j,L} - s_{j,R}) \sum_{i \neq j} b_\tau(i,j)\right\} \qquad (20)$$

Without loss of generality, let us assume that $(s_{j,L} - s_{j,R}) \geq (s_{j+1,L} - s_{j+1,R})$ for $1 \leq j < N$. If we can show that $b_{Swap}(i, j) \geq b_{MTE}(i, j)$ for $1 \leq j < i \leq N$, then by (20) our theorem is proved. Consider

$$\frac{b_{Swap}(i, j)}{b_{Swap}(j, i)} = \frac{\sum_{k's} \text{Prob}(R_{k_1} \ldots R_{k_x} R_i R_{k_{x+1}} \ldots R_{k_y} R_j R_{k_{y+1}} \ldots R_{k_{N-2}})}{\sum_{k's} \text{Prob}(R_{k_1} \ldots R_{k_x} R_j R_{k_{x+1}} \ldots R_{k_y} R_i R_{k_{y+1}} \ldots R_{k_{N-2}})}.$$

In the above, the sums are taken over all permutations $k_1, \ldots, k_{N-2}$ of the integers 1, 2,..., i-1, i+1,..., j-1, j+1,..., N. By using Theorem III this yields :

$$\frac{b_{Swap}(i, j)}{b_{Swap}(j, i)} \geq \min_{1 \leq x < y \leq N} \frac{\text{Prob}(R_{k_1} \ldots R_{k_x} R_i R_{k_{x+1}} \ldots R_{k_y} R_j R_{k_{y+1}} \ldots R_{k_{N-2}})}{\text{Prob}(R_{k_1} \ldots R_{k_x} R_j R_{k_{x+1}} \ldots R_{k_y} R_i R_{k_{y+1}} \ldots R_{k_{N-2}})}$$

$$= \min_{1 \leq x < y \leq N} \left[\frac{s_{i,L} + s_{j,R}}{s_{j,L} + s_{i,R}}\right]^{y-x}. \qquad (21)$$

However, $(s_{j,L} - s_{j,R}) \geq (s_{j+1,L} - s_{j+1,R})$ implies that $\frac{s_{i,L} + s_{j,R}}{s_{j,L} + s_{i,R}} \geq 1$. Clearly, the minimum of the R.H.S. of (21) is attained when y-x is unity, and thus,

$$\frac{b_{Swap}(i, j)}{b_{Swap}(j, i)} \geq \min_{1 \leq x < y \leq N} \left[\frac{s_{i,L} + s_{j,R}}{s_{j,L} + s_{i,R}}\right]^{y-x} = \frac{s_{i,L} + s_{j,R}}{s_{j,L} + s_{i,R}} \qquad (22)$$

But, $b_{Swap}(i, j) = 1 - b_{Swap}(j, i)$. Substituting this fact into (22) we get,

$$b_{Swap}(i, j) \geq \frac{s_{i,L} + s_{i,R}}{s_{j,L} + s_{i,R}}(1 - b_{Swap}(i, j))$$

$$\Rightarrow \quad b_{Swap}(i, j) \geq \frac{s_{i,L} + s_{j,R}}{s_{i,L} + s_{i,R} + s_{j,L} + s_{j,R}}$$

$$= b_{MTE}(i, j)$$

and the result is proved. •••

Observe that this result is a generalization of the result derived for the Transposition heuristic. The latter can be obtained by considering the special case in which $s_{i,R}=0$ for all i. But this is definitely not derivable as a direct consequence of the previously known results.

IV. EXPERIMENTAL RESULTS

To verify the results of Theorem II and Corollary II.1, we compared the MTF and Transposition heuristics under the undirected mapping. The probability p_i obeyed the Zipf distribution, which is,

$p_1 = C/1$, $p_2 = C/2$, ..., $p_N = C/N$,

where C is the normalizing constant and $p_{i,L} = p$ for i=1..N, with $p > 0.5$. To ensure the accuracy of the experiments, 100 tests of 7000 queries each were conducted and the results presented are the ensemble average costs of the last 500 queries. Figure I shows the results of the simulations. Notice that the performance of two scheme is exactly the same when p = 0.5. The Transposition heuristic under undirected mapping executes progressively better than the MTE heuristic under undirected mapping as p increases.

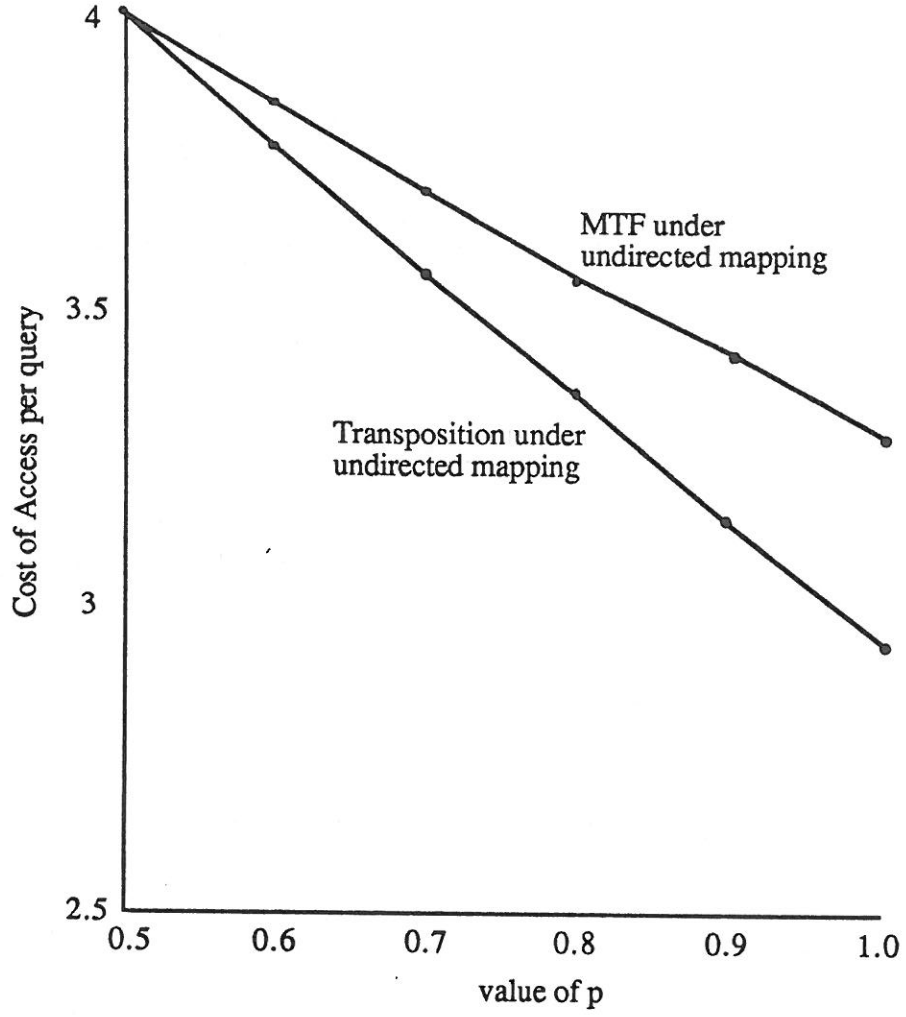

Figure 1. Cost comparison between the MTF and Transposition schemes under an undirected mapping. There are seven elements in the list and p is the probability that an element is accessed from the left.

To demonstrate the power of Theorem III, various experiments were also done to compare the performance between Move-To-End and the Swap heuristics. The environment was simulated by two Zipf distributions in which $R_1, R_2, ..., R_N$ are accessed with decreasing probability from the left end of the list but are accessed with increasing probability from the right end of the list. That is,

$$s_{1,L} = C/1,\ s_{2,L} = C/2,\ ...,\ s_{N,L} = C/N,$$
$$s_{1,R} = C/N,\ s_{2,R} = C/(N-1),\ ...,\ s_{N,R} = C/1, \qquad (23)$$

with C being a normalizing constant.

Different values of N were used to observe the changes in the average access cost. To ensure that a good estimate of the asymptotic cost of the two schemes was obtained for

various values of N, the lists were allowed to converge by increasing the number of queries per test to 1000N. Again, the ensemble average of the last 500 queries are surveyed for each of the 100 experiments conducted. The details of the simulations are given in Table I and a comparative survey of the heuristics compared to the optimal list arrangement is given in Table II.

Elements	Move-To-End	Swap	Optimal
7	3.32656	2.9797	2.6997
10	4.34282	3.71238	3.4142
15	5.84622	4.814	4.5205

Table 1. Cost comparison between the MTE and Swap heuristics and the Optimal list arrangement under the Zipf's distribution defined by (23).

Elements	Move-To-End	Swap
7	23.22	10.37
10	27.20	8.734
15	29.33	6.493

Table 2. Percentage increase in cost of the MTE and Swap heuristics compared to the Optimal list arrangement under the Zipf's distribution defined by (23).

From the Tables above, observe that when N=7, the Swap heuristic has an average cost of 2.9797, which is 10.37% more expensive than the Optimal arrangement; on the other hand, the MTE heuristic has an average cost of 3.3266, which is 23.22% above the cost of the Optimal arrangement. In general, this percentage ratio decreases for the Swap heuristic but increases for the MTE heuristic with respect to N.

To conclude this section, we conjecture that if the singly linked list heuristics τ is more efficient than ρ, then τ', the doubly linked list heuristic obtained using the directed

mapping on τ, is also more efficient than ρ', the corresponding doubly linked list heuristic obtained using the directed mapping on ρ. This conjecture is an generalization of both Theorem II and Theorem IV for the set of directed mappings. It seems rather intuitive since a more efficient scheme is likely to have a tendency to push the frequently accessed elements to the end, and thus plausibly do a better job to polarize the elements in the doubly linked list data structure. However, we have not yet proved the result.

5. CONCLUSION

In this paper, we have studied the problem of adaptively reorganizing doubly linked lists. To compare various underlying user query patterns we have introduced the concept of stochastic completeness. The concept has been used to compare various representations of the same underlying environment. Using this concept, we have presented two interchangeable representations which can be used to express independent and time invariant doubly linked list environments. We have also presented interesting relationships between singly linked list heuristics and doubly linked list heuristics by the use of undirected and directed mappings. In particular, we have introduced a new heuristic called the Swap heuristic, and have shown it to be more efficient than the MTE heuristic. We are currently investigating the use of the adaptive doubly linked list structures and the adaptive circular list to study environments in which the query distributions are time invariant but **dependent**.

REFERENCES

[Ande85] Anderson, E.J., Nash, P. and Weber, R.R., A counter-example to a conjecture on optimal list ordering, J. Appl. Probl. 19, 3 (Sept.82), pp. 730-732.

[Bitn79] Bitner, J.R., Heuristics that dynamically organize data structures, SIAM J. Comput. 8, 1 (Feb.79), pp. 82-110.

[Gonn79] Gonnet, G.H., Munro, J.I., and Suwanda, H., Toward self-organizing linear search, Proceedings of the 20th IEEE Symposium on Foundations of Computer Science (San Juan, Puerto Rico, Oct.79). IEEE, New York, pp.169-174.

[Hest85] Hester J.H. and Hirschberg D.S., Self-organizing linear search, Comp. Surveys, Vol. 17, 3 (Sept.85), pp. 295-311.

[Hend72] Hendricks, W.J., The stationary distribution of an interesting Markov chain, J.Appl.Probl. 9, 1 (Mar.72), pp. 231-233.

[Kan80] Kan, Y.C. and Ross, S.M., Optimal list order partial memory constraints, J. Appl. Prob. 17, 4 (Dec. 80), pp. 1004-1015.

[Knut73] Knuth, D.E., The art of computer programming, vol. 3: Sorting and Searching. Addison-Wesley, Reading, Mass.(1973), pp.398-399.

[Math80] Matthews, D., Self-organizing doubly linked lists, J.Comp. Maths., Sec.A, Vol.8 (1980), pp. 99-106.

[McCa65] McCabe, J., On serial files with relocatable records, Oper. Res. (July./Aug. 65), 609-618.

[Oomm87] Oommen, B.J., and Hansen, E.R., List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations, SIAM Journal of Computing, Vol. 16, No.4, (Aug. 1987), pp 705-716.

[Oomm88] Oommen, B.J., Hansen, E.R. and Munro, J.I., Deterministic optimal and expedient move-to-rear list organizing strategies, to appear in Theoretical Computer Science.

[Rive76] Rivest, R., On self-organizing sequential search heuristics, Comm. ACM 19, 2 (Feb.76), pp. 63-67.

[Trem76] Tremblay, J.P., and Sorenson, P.G., An introduction to data structures with applications, McGraw-Hill, New York, 1976.