Move-To-End is best for Double-Linked Lists

Vladimir Estivill-Castro
LANIA, Rébsamen 80, Xalapa, Veracruz, 91000, México
vlad@xalapa.lania.mx

Abstract

We demonstrate that

- Move-To-End (ME) is statically competitive for
doubly linked lists. That is, ME is competitive
with respect to the class of off-line static heuristic
for doubly-linked lists.

- ME is competitive with respect to the class of on-
line dynamic heuristics that use, per query, one
end of the doubly-linked list.

Moreover, other heuristics, like Swap do not have
these properties. Since we prove that there is no
competitive heuristic for the class of on-line dynamic
memoryless deterministic heuristics, ME offers the
best competitiveness behavior.

1 Introduction

Self-organizing lists and self-organizing trees are
dynamic data structures that attempt to minimize
the cost of dictionary operations by promoting high-
frequency elements to easily accessible locations. In
many applications of dictionaries the distribution that
governs the operations is difficult to determine a priori
or the probabilities fluctuate in time. However, self-
organizing data structures adapt to the current distri-
bution, take advantage of nonuniform access proba-
bilities and profit from locality of reference. The sim-
plest and most obvious structure for a dictionary is
a sequential list. Researchers have devoted extensive
efforts to the design and analysis of heuristics for self-
organizing sequential lists. A comprehensive lists of
references can be found in Hester and Hirchberg's sur-

In this paper we are interested in studying self-
organizing heuristics for double-liked lists — a more
sophisticated structure than sequential lists, but still
one dimensional. The study of self-organizing double-
linked lists was initiated by Matthews, Rotem and
Bretholz [5] and has recently acquired renewed in-
terest [6,9]. However, the work in the literature has
focused only on heuristics that attempt to optimize
the asymptotic expected search time. This evaluation
criteria ignores the benefits of strategies that profit
from locality of reference and it does not take into ac-
count other aspects like the total cost of a sequence of
requests. We present competitiveness results for the
Move-to-End (ME) heuristic for double-linked lists. In
particular, we will describe three models of queries for
a double-linked list and determine that for the first
two models, ME is competitive, while for the third,
no competitive heuristic is possible. This illustrates
that ME offers the best competitive behavior.

The paper is organized as follows. In Section 2 we
introduce notation and definitions. In Section 3 we de-
define the abstract data type Double-Linked List (DDL).
Section 4 presents competitiveness results for ME. We
conclude with some final remarks in Section 5.

2 Definitions

A sequential list is a list of records that will be
searched sequentially on the basis of a key value as-
sociated with each record. Usually, the records are
accessed with nonuniform probabilities and/or the se-
quence of requests presents some locality of reference.

Two formalisms have been developed to evaluate
the relative merits of self-organizing heuristics for lists.
The first consists of comparing the asymptotic ex-
pected search cost, while the second consists of com-
paring the cost of processing a sequence of opera-
tions and presenting competitiveness results. The
former was initially developed because, after several
operations, self-organizing heuristics reduce the ex-
pected search cost [4,7]. However, the analysis of
the expected cost does not determine the heuristic to
use in practice. For example, although Transpose's
(TR) asymptotic expected cost outperforms Move-to-
Front's (MF) [7], it turns out that it takes longer for
TR to reach a state close to optimum and practical evidence has motivated some researchers to recommend MF over TR for linked list implementations [2]. This observation found rigorous justification on the analysis of the cost incurred by a self-organizing heuristic on a sequence of operations [1, 8]. More importantly, the assumptions required to carry out the analysis of the asymptotic expected search cost may not always hold whilst the amortized analysis of the cost incurred over a sequence of operations is a reliable criteria to evaluate the performance of the heuristic. If \( \sigma = \sigma_1, \ldots, \sigma_m \) is a sequence of queries to a list, the cost of servicing \( \sigma \) under heuristic \( H \) is the sum of the costs of serving each \( \sigma_i \) and is denoted by \( \text{cost}_H(\sigma) \). The power of an heuristic with respect to a family of alternatives is evaluated through the notion of competitiveness.

Definition 2.1 With respect to a class of heuristics \( \mathcal{H} \), the competitive ratio of an heuristic (algorithm) \( H \) is the supremum of \( \text{cost}_H(\sigma)/\text{cost}_{H'}(\sigma) \), where \( H' \) is any heuristic in the class \( \mathcal{H} \) and \( \sigma \) is any sufficiently long sequence of queries. \( H \) is \( \alpha \)-competitive if its competitive ratio is no more than \( \alpha \). \( H \) is competitive if its competitive ratio is a constant.

The following are commonly used criteria to classify heuristics according to the power of the transformations they can perform in the data structure or their look-ahead into the sequence of requests.

- A heuristic \( H \) for maintaining a data structure \( D \) is static if \( H \) does not modify \( D \) during the processing of any sequence of requests (note that \( D \) may be preprocessed once before processing the sequence). On the other hand, a heuristic \( H \) for maintaining a data structure \( D \) is dynamic if, for a sequence of requests \( \sigma \), the heuristic \( H \) may need to modify \( D \) during the processing of a query in \( \sigma \).
- A heuristic \( H \) is said to operate on-line if the processing of a query is performed without any knowledge of future queries. On the other hand, an heuristic \( H \) is said to operate off-line if the processing of a query uses information about future queries. Queries must be processed in the order they are presented.

3 Double-linked lists

A self-organizing heuristic for a double-linked list receives queries of the form \text{retrieve}(\text{key}, \text{window}) \) where "key" is the key of the record to be searched and "window" is a reference (pointer) to one of the two ends of the double-linked list. The heuristic is allowed to construct several windows while processing a query but we will place limitations on the windows that the heuristic can remember between processing of queries. The heuristic can not create copies of the double-linked list and the heuristic has access to the double-linked list only through the following operations, all of which cost constant time:

- \text{Compare}(k, w) \) — Returns TRUE if the key in the record pointed by \( w \) and \( k \) are equal, FALSE otherwise.
- \text{MoveLeft}(w) \) [\text{MoveRight}(w) \) — if \( w \) points to a record that has a record \( \rho \) to its left [right], then \text{MoveLeft} \) [\text{MoveRight} \) returns TRUE and \( w \) is updated to point to \( \rho \); otherwise \text{MoveLeft} \) [\text{MoveRight} \) returns FALSE.
- \text{Remove}(w, \rho) \) — the information in the record pointed by \( w \) is copied into \( \rho \) and the records pointed by \( w \) is removed from the data structure. All other elements preserve their relative order.

4 Competitiveness results

We now show that \( ME \) is statically competitive for doubly-linked list. We also show that \( ME \) is dynamically competitive with respect to the class of heuristics that use only one end of the doubly-linked list at a time, but it is not competitive for heuristics that
The heuristic ME is competitive with respect to the class of off-line static heuristics for doubly-linked lists.

Sketch of the Proof: Let $H$ be the class of off-line static heuristics for doubly-linked lists. Let Decreasing Frequency (DF) be the following algorithm: For a sequence $\sigma$ of request, analyze $\sigma$ off-line and let $k_i.L$ $(k_i.R)$ be the number of accesses to record $x_i$ from the left (right), for $1 \leq i \leq n$. DF initializes the doubly-linked list, from left to right, in descending order of the values $k_i.L - k_i.R$. DF organizes the doubly-linked list before serving $\sigma$ and keeps this ordering through the processing never modifying the doubly-linked list any more.

It is not hard to show that DF is in $H$ and competitive with respect to $H$ with optimum competitive ratio. Thus, to prove the theorem it is enough to show that ME is competitive with respect to DF.

The rest of the proof is a potential function argument common in this type of results.

The following theorem describes the family of heuristics for doubly-linked list for which ME is dynamically competitive.

Theorem 4.2 Let $H$ denote the class of dynamic heuristics for doubly-linked lists that, while processing a query, use only the end of the doubly-linked list that is provided with each query. Then ME is competitive with respect to $H$.

Sketch of the Proof: Consider $\sigma$ a sequence of request on doubly-linked lists representing a set of records $X$. Let $H$ be the family of dynamic heuristics for doubly-linked list that use only the window in the query to serve each query. Let $H \in H$. We use a potential function argument. Let $\phi(j)$ be the number of inversions between the ordering of $H$'s doubly-linked list and the ordering of ME's doubly-linked list after both have served the $j$-th query. Let $t_j$ denote the actual cost incurred by $ME$ when serving the $j$-th query, let $a_j$ denote the amortized cost of serving the $j$-th query and defined by $a_j = t_j + \phi(j) - \phi(j-1)$. If $a_j = 2[\phi(H(\sigma_j))] - 1$: for all $j > 0$, then,

$$
\text{cost}_{ME}(\sigma) = \sum_{j=1}^{\text{|}\sigma\text{|}} t_j
\leq 2[\phi(H(\sigma))] - |\sigma| + \phi(0) - \phi(|\sigma|),
$$

and the results follows. Thus, we only need to prove Equation (1). Suppose $\sigma_j$ is the $j$-th query in $\sigma$. There are two cases:

Case 1: The operation is an access from the left. Suppose that $x_i$ is the $k$-th request from the left in ME's list. The actual cost of the retrieval of $x_i$ by $ME$ is $k$. Now $H$ searches $x_i$ from its left end, suppose $H$ finds $x_i$ in the $h$-th position and after the search it places $x_i$ in its list $e_j$ places left or right from the $h$-th position. If ME move $x_i$ left, we charge $H$ only $h$, but if it moves further right, we charge $h + e_j$ units. We now calculate the change in potential. Say $k = I + N$, where $I$ is the number of elements in the first $k - 1$ positions of ME's list that are inverted with $x_i$ in $H$ list, and $N$ is the number of elements in the first $k - 1$ positions of ME's list that are not inverted with $x_i$ in $H$ list. After ME moves $x$ to the left end, we have $I$ inversions less and $N$ new inversions. Since $N$ elements are not inverted with $x_i$ at the beginning of the $j$-th operation, they also appear before $x_i$ in $H$'s doubly-linked list. Thus, $N = k - I - 1 \leq h - 1$. Using Iverson's notation $1$ it is not hard to prove that

$$
a_j = k + (N - I) - (H \text{ move } x_i \text{ left})e_j
\leq 2[\phi(H(\sigma_j))] - 1.
$$

Case 2: The operation is an access from the right. This is symmetric to Case 1 above.

This completes the proof.

However, ME is no longer competitive with respect to $B$ — the family of heuristics that can access the doubly-linked list from both ends for each query. Consider the following example. Let $X$ be a set where $\|X\| > 4$ and let $x \in X$. Consider the sequence of queries $\sigma = ([x, \text{left}), (x, \text{right})]^m$ which consists of queries for the same records but alternating the end of the doubly-linked list. Clearly, $\text{cost}_{ME}(\sigma)$ is $\Theta(\|X\| |\sigma|)$. However, the heuristic Move

$1$For a predicate $P$ let $(P) = 1$ if $P$ is TRUE, $(P) = 0$ otherwise.
Back (MB) that after searching for a record places it at the front of the other end to where the search was started is such that \( \text{cost}_{MB}^B(\sigma) = O(|\sigma|) \). Since \( \text{cost}_{MB}^B(\sigma) / \text{cost}_{MB}^B(\sigma) = \Omega(|X|) \), MB is in \( B \) and it is even on-line, we obtain that ME is not dynamically competitive with respect to \( B \).

It seems disappointing that ME is not competitive with respect to on-line memoryless dynamic heuristics that have windows to both ends of the doubly-linked list. However, we now show that there is no competitive algorithm for this class when the search must be started at the end the query commanded.

We now rigorously define the notion of memoryless and the notion of cost enforced by the query.

Definition 4.1 A deterministic memoryless heuristic for doubly-linked lists is described by a sequence of permutations \( \tau = \{\tau^L_i, \tau^R_i\}_{i \in \mathbb{N}} \). The permutation \( \tau^L_i \) is applied whenever the element is located by a search from the left at the i-th position. The permutation \( \tau^R_i \) is applied whenever the element is located by a search from the right at the i-th position.

Note that a reorganizing permutation is applied after the search and it depends only on the direction of the current search and the position where the record is found.

Definition 4.2 A heuristic for doubly-linked lists is said to have cost enforced by the direction of the search if the cost of a query retrieve \( (k, w) \) is at least the distance from the position of the record \( k \) to the end \( w \).

Thus, a heuristic with cost enforced by the direction may use the other end of the doubly-linked list to re-arrange the list; however, it will always be charged for a search at least the cost implied by the query.

Theorem 4.3 Let OMDD be the class of On-line Memoryless Dynamic heuristics with cost enforced by the Direction of the search. There is no competitive heuristic in OMDD with respect to OMDD.

Sketch of the Proof: Let \( H \) be any heuristic in OMDD. Let \( X \) be a set with \( n \) elements represented in a doubly-linked list. Let \( x \in X \). For a positive integer \( k \), we denote by \( \sigma^k(x) \) the following sequence of requests: \( \sigma^k(x) = (\{x, left\} \cup \{x, right\})^{k}\bmod{2^k} \), where \( m \) is long enough and its the length of \( \sigma^k(x) \). The proof consists of showing that there is a family of heuristics \( \{H^k(x)\}_{x \in \mathbb{N}} \) in OMDD such that \( \text{cost}_{H^k(x)}(\sigma^k) = O(|\sigma^k|) = O(m) \). Thus, if \( H \) is competitive in OMDD, then \( \text{cost}_H(\sigma^k) \) must be a constant factor away from \( \text{cost}_{H^k(x)}(\sigma^k) \) for each \( k \). Therefore, \( \text{cost}_H(\sigma^k(x)) = O(m) \) for each \( k \) and each \( x \). But since \( H \) is memoryless, this is not possible.

5 Final remarks

Move-to-Front is dynamically competitive even with respect to off-line heuristics for sequential lists. Moreover, when generalized to the environment of paging as the "least recently used" rule, it results in an algorithm with a competitive factor that depends on the size of fast memory and no on-line algorithm has better performance. Also, Splay-trees have a self-organizing rule corresponding to a generalization of Move-to-Front, they are statically competitive and believed to be dynamically competitive with respect to binary trees. Move-to-End is the directed generalization to doubly-linked lists of Move-to-Front. Our results show that ME generalizes the competitiveness of Move-to-Front as much as it is possible for doubly-linked lists.

References