ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 2, Issue 12, December 2013

Experimental Analysis of Novel Variant of MFK Algorithm

Debashish Rout
Department of Computer Science and Engineering,VSSUT,Burla,India

1. Introduction

Data structure provides a way to store data in structure way efficiently, in the primary memory of computer. Various operations such as search, insert and delete can be performed on a data structure. If the data items are unsorted and stored in a linear list, each item can be searched by scanning the list of items one by one linearly. In *linear search* if multiple elements are searched, the total search time can be reduced by making the data structure self-organizing. In self-organization data structure the items can be re-organized after each operation to reduce the time of future operations Thereby enhancing the performance.

1.1 List Accessing Problem

List Accessing problem or List Update problem is the method used in the self-organizing linear search. In List Update problem a list (l) of records and a request sequence (σ) are taken as inputs. When a record is accessed from the list then the list is reconfigured to reduce the future search cost. When a record is accessed from the list, some cost is provided for that. List accessing problem is mainly implemented by single linked list. But it may be implemented through doubly linked list and tree also.

1.2 Cost Model

In the list accessing problem, we have two different models based on operations and list type. They are Static list accessing model and Dynamic list accessing model. The Static list accessing model is the one in which the number of items in the list is fixed and only the access operation can be performed. The Dynamic list accessing model is the one in which the size of the list varies dynamically and all the three operations i.e. insert, delete and access can be performed. In our work, we have considered only the static model of list accessing problem and hence we consider only the access operation. As one of the key issues is to find out the optimal access cost of elements on the list, we need a cost model which is an efficient tool to measure the access cost incurred by the various list accessing algorithms. A number of cost models have been developed and used so far but here we have considered only Full Cost Model (FCM) and Partial Cost Model (PCM). In Full Cost Model, the cost of accessing an item in the i^{th} position from the front of the list is i. In the Partial Cost Model the cost of accessing an item in the i^{th} position from the front of the list is (i-1) because we have to make (i-1) comparisons before accessing the i^{th} element in the list. So, the cost of accessing the first element in the list would be 1 in FCM and 0 in PCM.We are illustrating both the models as follows. Suppose the list is 1, 2, 3 and the request sequence is 1, 2, and 3. The costs of elements according to the various models are presented in below.

1.3 Application

List accessing algorithms are widely used in Data Compression. Other important applications of list update algorithms are computing point maxima in computational geometry, resolving collisions in hash table and dictionary maintenance. The List Accessing Problem is also of significant interest in the contest of self organizing data structures.

Elements	Access cost in PCM	Access Cost in FCM
1	0	1
2	1	2
3	2	3
Total cost	3	6

1.4 List Accessing Algorithms

The algorithm which efficiently recognizes the list and reduces the cost for the access is called a list accessing algorithm. List accessing algorithms can be of two types such as online algorithm and offline algorithm. In online algorithm, the request sequence is partially known. In offline algorithm, the request sequence is fully known; online algorithms can be further classified into two types such as deterministic and randomized. Deterministic algorithm is one which produces the same output always for a given request sequence or the algorithm passes through same states for a given request sequence. Some of the popular deterministic algorithms for the list accessing problem are Move-To-Front (MTF), Transpose (TRANS) and Frequency Count (FC).

MTF: After accessing an item, it is moved towards the front of the list without changing the order of other items in the list.

TRANS: After accessing an element, it is exchanged with its proceeding element.

FC: There is a counter for each item which counts the frequency of each item of the list according based on the requests from the request sequence. The list is arranged in the non-increasing order of frequency count of items in the list. In randomized online algorithm, while processing the request sequence, the algorithm makes some random decision at some step. Some well known randomized algorithms are SPLIT, BIT, COMB and TIME-STAMP.

1.5 Literature Review

List update problem was first studied by McCabe [3] in 1965 with the concept of relocatable records in serial files. He also introduced two list accessing algorithms Move-To-Front (MTF) and Transpose (TRANS). Rivest has examined a class of heuristics for maintaining the sequential list in optimal order with respect to the average time required to search for a specified element with an assumption of fixed probability of each search in his experimental study.He has shown that MTF and Transpose heuristic are optimal within a constant factor. Hester and Hirschberg have done a comprehensive survey of all permutation algorithms that modified the order of linear search lists with an emphasis on average case analysis. Sleator and Tarjan in their seminar paper have formally introduced the concept of competitive analysis for online deterministic list update algorithms such as MTF, TRANS and FC using amortized analysis and potential function method. MTF is proved to be 2-competitive where as FC and TRANS are not competitive. Irani proposed first randomized online list update algorithm known as SPLIT which is 1.932-competitiv. Albers, Von-Stengel, and Werchner proposed a simple randomized online algorithm-COMB that achieves a 1.6-competitive, the best randomized algorithm in literature till date. Albers introduced the concept of look ahead in the list update problem and obtained improved competitive ratio for deterministic online algorithms. Reingold and Westbrook have proposed an optimal offline algorithm which runs in time $O(2^l l! n)$ where l is the length of the list and n is the length of request sequence. Bachrach et al. have provided an extensive theoretical and experimental study of online list update algorithm in 2002. The study of locality of reference in list accessing problem was initiated by Angelopoulos in 2006, where he proved MTF is superior to all algorithms. Relatively less work has been done on the offline algorithms for the list accessing problem. For analyzing the performance of online algorithm by competitive analysis an optimal offline algorithm is essential. Ambühl in 2000 proved that off-line list update is NP-hard by showing a reduction from the Minimum Feedback Arc Set Problem. In 2004, Kevin Andrew and David Gleich showed that the randomized BIT algorithm is 7/4-competitive using a potential function argument. They introduced the pair-wise property and the TIMESTAMP algorithm to show that the COMB algorithm, a Combination of the BIT and TIMESTAMP algorithms, is 8/5-competitive. In 2009, in one of the paper a survey has been done on online algorithms for self organizing sequential search . Some recent works on the LUP have been done in .

1.6 Our Contribution

In this paper, we have proposed a novel variant of MFK algorithm, which we popularly call as VMFK.We have also performed empirical study and comparative performance analysis of VMFK with MTF using three data sets such as Calgary Corpus, Canterbury Corpus. Our experimental results show that VMFK outperforms for all request sequences for the two datasets, Calgary Corpus and Canterbury Corpus.

1.7 Organization of paper

The paper has been introduced in section 1.MFK algorithm is discussed in section 2. Proposed algorithms are discussed in section 3. Section 4 shows experimental analysis of the algorithm. The paper is concluded in section 5 followed by a set of references

2. MFK Algorithm

For our experimental analysis we have considered different types of request sequences as input to different existing List accessing algorithms and proposed variants of MTF algorithm. For each algorithm we have generated the request sequence and list from different text files of Calgary Corpus and Canterbury Corpus dataset. The total access cost of each algorithm is calculated for different request sequences and list. Then we have compared the access cost of existing algorithm and our proposed algorithms. We have proposed a novel idea to design a new List Accessing Algorithm to reduce the future Access Cost based upon the MFK algorithm i.e. Variant MFK Algorithm (VMFK).

MFK Algorithm: *Upon an access of item x in the list, move x to the front of the list if it has been requested k **(exactly k)** times in a row under a number of interesting probability distributions. Else list remains unchanged*.

3. Our proposed Algorithms

3.1 Concept and Ideas

VMFK Algorithm: *Upon an access of item x in the list, move x to the front of the list if it has been requested atleast k **(>k)** times in a row under a number of interesting probability distributions. Else list remains unchanged.*

3.2 Pseudo Code

Inputs:
l: size of the List L
n: size of the request sequence σ
Outputs:
$C_{MFK:}$ *Cost of MFK Algorithm*
Notations:
$P_{i:}$ *Position of* i^{th} *item in the list,* $1 \le i \le l$
σ_j*:-*j^{th} *scanned item in the request sequence,* $1 \le j \le n$
$C(\sigma_j)$*: Access cost* σ_j *in the list L*
$F(\sigma_j)$*:-Frequency of* σ_j *the request sequence σ before accessing the item in the list*

Algorithms

```
Initialize C_MFK=0;
k=1;
for j=1 to n
  {
   read the request σ_j in the σ;
   Scan σ_j in the L;
      x=σ_j;
   Let P_i be the position of x in L
     C(x) =P_i;
     C_MFK=C_MFK +C(x);
   if F(x) >k
        move x to the front of L
   else
      L remains unchanged
   }
```

3.3 Illustration of Variant of MFK Algorithm

- In VMFK algorithm k is independent of list size (l).k varies between $1 \le k \le RS - LS + 1$
  Where RS stands for Request Sequence Size and LS stands for List Size.
- k is fixed for all the Request sequence generated from the dataset.
- For our experiment we have taken k=1 with suitable probability distribution.

Significance of VMFK: List doesn't change always.
Let's consider the list 12345 and the request sequence is 11234453.When 1 is accessed it is present in the first position of the list and the cost is 1.As before accessing the element 1 from the.

4. Experimental Analysis

4.1 Experimental Setup

The proposed algorithm VBIT is tested with respect to two large well known datasets called as Calgary Corpus and Canterbury Corpus, which are extensively used for data compression. We performed 4 experiments. The goal of the first experiment was to remove all the spaces from the text files of Calgary Corpus and Canterbury Corpus. The second experiment was to obtain distinct characters from the file created through first experiment and this file will be used as input both for list and request sequence. The third and fourth experiment was to implement BIT and VBIT respectively. The source code is implemented through "MATLAB 7.10.0(R2010a)" and windows environment. RAM size is 1 GB and processor speed is 1.80 GHz.

4.2 Input Dataset

The Calgary corpus is a collection of (mainly) text files that serves as a popular bench mark for testing the performance of (text) compression algorithm and can also be used for access cost performance testing. The corpus contains 9 different types of files and overall 17 files. In particular it contains books, papers, numeric data, pictures, programs and object files. This was developed in the late 1980s, and during the 1990s became something of a de facto standard for lossless compression evaluation. The collection is now rather dated, but it is still reasonably reliable as a performance indicator. It is still available so that older results can be compared.
Canterbury Corpus collection is the main benchmark for comparing compression methods. The Calgary_collection is provided for historic interest, the Large corpus is useful for algorithms that can't "get up to speed" on smaller files, and the other collections may be useful for particular file types. This collection was developed in 1997 as an improved version of the Calgary corpus. The files were chosen because their results on existing compression algorithms are "typical", and so it is hoped this will also be true for new methods. There are 11 files in this corpus.
Each file was used to generate 2 different request sequences. The first sequence was generated by parsing the files into "words" ('word' parsing). A word is defined as the longest string of non space characters. For some of the non text files in the corpus (e.g. pic), the parsing does not yield a meaningful sequence, hence results are ignored. The second sequence is generated by reading the file as a sequence of bytes (Byte Parsing).

4.3 Experimental Performance

The input to each algorithm is a byte parsing of each of the file. The LS is created when the RS are parsed. A table is created for byte parsing. The table contains the number of items in the request sequence and the number of items in the list sequences that are generated for each file. The cost of MTF and VMFK are computed and recorded for each file as shown in table.

4.4 Experimental Results

For our experiments we have considered bytes and words of the file as request sequence and list. In order use files for our experiment we have used the files of Calgary Corpus and Canterbury Corpus. From the files of Calgary corpus and Canterbury Corpus, with the help of our program ***main_MTFvsMund_greaterthanK.m*** we have omitted all the spaces used in the Calgary corpus file and Canterbury Corpus and it is represented in the following table.

Dataset	|LS|	|RS|	C_{MTF}	C_{VMFK}
book1	**81**	**768771**	**9770030**	**9769028**
bib	**81**	**111261**	**2197756**	**2195381**
geo	**255**	**102400**	**4811399**	**4800477**
news	**98**	**377109**	**6409661**	**6408233**
progp	**89**	**49379**	**716228**	**715608**
progl	**87**	**71646**	**871903**	**870547**
paper1	**95**	**53161**	**781555**	**780983**
paper3	**84**	**46526**	**614682**	**614255**
obj1	**255**	**21504**	**707249**	**702639**

Table 4.1.1 Access cost incurred by MTF and VMFK for Calgary Corpus with Byte Parsing

Dataset	|LS|	|RS|	C_{MTF}	C_{VMFK}
alice29	74	152089	2025917	2025057
asyoulik	68	125179	1908414	1907654
lcet10	84	426754	5570790	5569463
Fields.c	90	11150	170270	169536
plrabn12	81	481861	6473411	6471937
ptt5	159	513216	1609438	1598901
grammar.lsp	76	3721	46330	46063
kennedy.xls	256	1029744	21557619	21556945
cp.html	86	24603	440325	438942

Table 4.1.2 Access cost incurred by MTF and VMFK for Canterbury Corpus with Byte Parsing

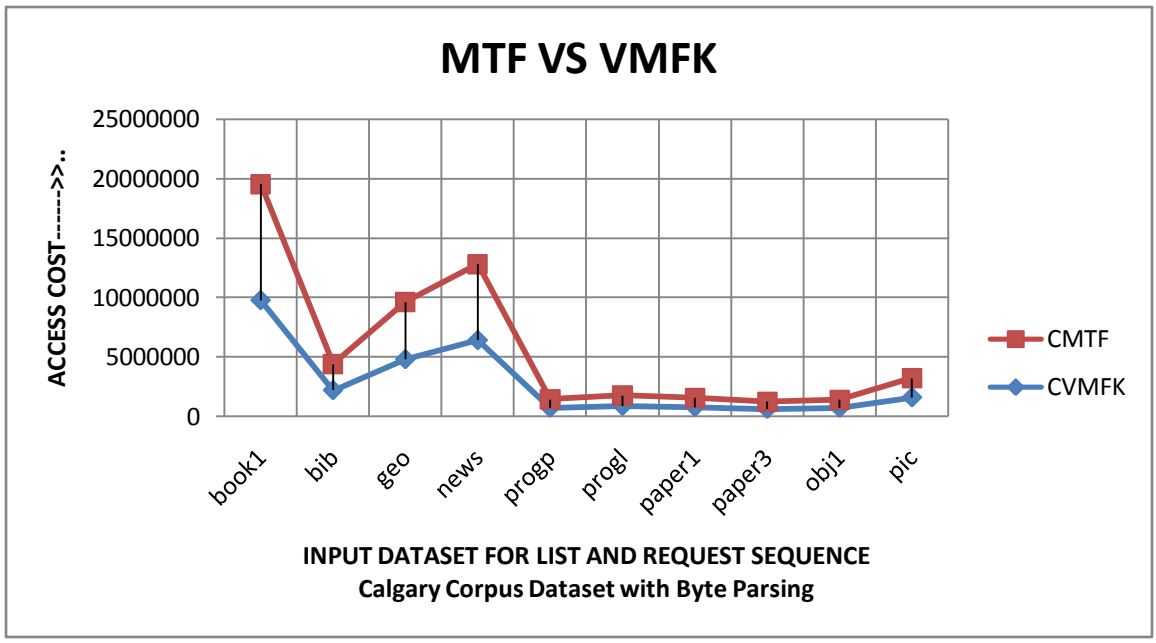

Figure 4.1.1 Cost incurred by MTF and VMFK against for Calgary Corpus

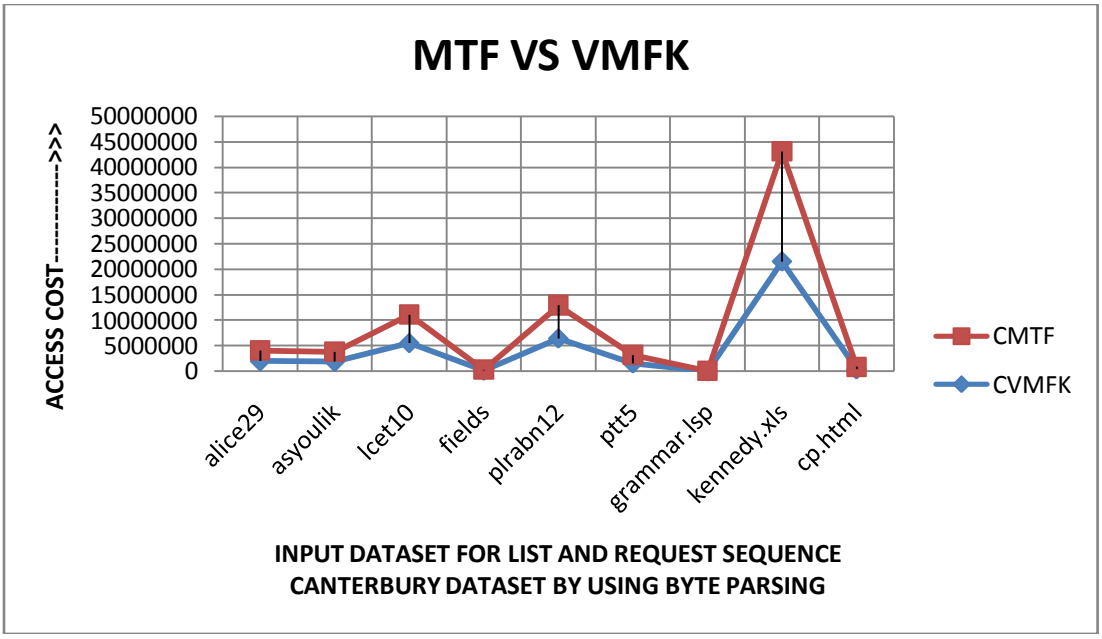

Figure 4.1.2 Cost incurred by MTF and VMFK against for Canterbury Corpus

Table 4.1.3 Access cost incurred by MTF and VMFK for Calgary Corpus with Word Parsing

Dataset	|LS|	|RS|	C_{MTF}	C_{VMFK}
book1	30531	125553	618040497	578625577
book2	21954	85886	298671587	294174324
bib	5020	13740	17456400	16722912
paper2	3736	12113	9368028	9161182
paper3	2701	6155	4453874	4441517

Dataset	|LS|	|RS|	C_{MTF}	C_{VMFK}
asyoulik	6539	19360	26676439	25583406
kennedy.xls	843	844	356256	355818
cp.html	892	1426	455889	446720

Table 5.1.4 Access cost incurred by MTF and VMFK for Canterbury Corpus with Word Parsing

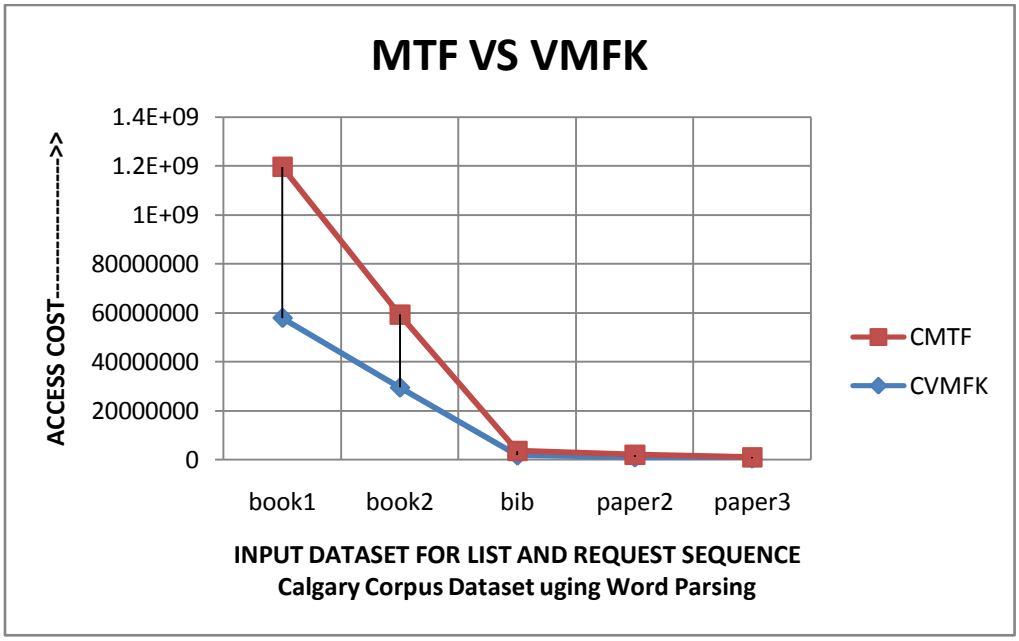

Figure 5.1.3 Cost incurred by MTF and VMFK against for Calgary Corpus

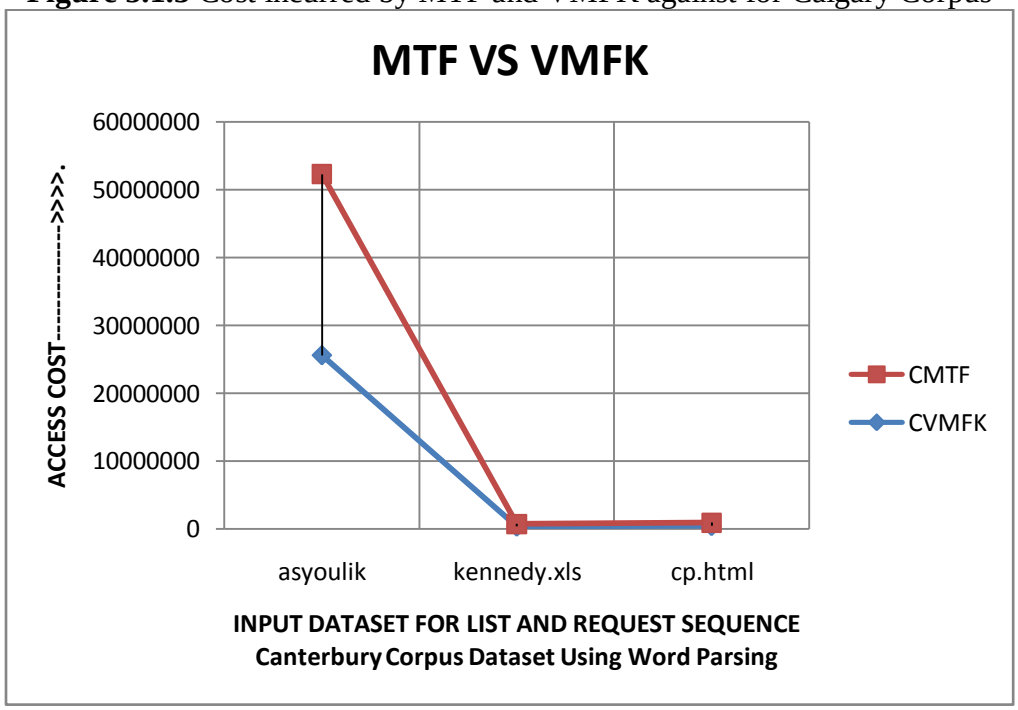

Figure 5.1.4 Cost incurred by MTF and VMFK against for Canterbury Corpus

5. Conclusion

List Accessing Algorithm take two parameter as input i.e. List and Request Sequence. Here we have proposed some novel variants MTF algorithm i.e. MPITF and MSITF and also some variants of MFK i.e. VMFK. Then we have made a cost comparison between MTF and MPITF, MTF and MSITF and MTF and VMFK. Then from that comparison we derive some theoretical results for MPITF and MSITF which is better than MTF .We have also implemented the proposed variants of MFK Algorithm. After implementing the MTF and VMFK algorithm through Calgary Corpus dataset, we have compared their accessed cost .From the comparison it is clear that VMFK is better than MTF algorithm.

Acknowledgement:-

This work is during my M.TECH career under supervision of Rakesh Mohanty, Lecturer, Department of Computer Science and Engineering, VSSUT, Burla.

Reference-

1. J. McCabe, “On serial files with relocatable records”, Operational Research, vol. 12, pp.609-618, 1965.
2. G. Jr. Schay, F.W. Dauer-“A Probabilistic Model for a Self Organizing File System”, SIAM J.of Applied Mathematics, (15) 874-888, 1967.
3. P.J Burvile and J.F.C. Kingman “On a model for storage and search” –J. of Applied Probability, 10:697-701, 1973.
4. R. Rivest-“On Self Organizing Sequential Search Heuristics”, Communications of the ACM, 19, 2:63-67, 1976.
5. J.R. Bitner” Heuristics that dynamically organize data structures”- SIAM J. of Computing, 8(1):82-110, 1979.
6. G.H. Gonet, J. I, Munro and H. Suwanda, “Towards self organizing linear search”-FOCS, 169-174, 1979.
7. G.H. Gonet, J. I, Munro and H. Suwanda, “Exegenesis of self organizing linear search”, SIAM Journal of Computing, Vol. 10, no.3, pp.613-637, 1981.
8. D. D. Sleator and R. E. Tarjan, “Amortized efficiency of list update paging rules”, Commun. ACM, vol. 28 no. 2, pp.202-208, 1985
9. J. H. Hester, D. S. Hirschberg” Self organizing linear search” – ACM Computing Surveys, 17(3): 295-312, 1985.
10. J.L. Bently and C. C. McGeoch, “Amortized analysis of self-organizing sequential search heuristics”, CACM, vol. 28, pp. 404-411, 1985.