E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

Experimental Analysis of Hybridized MTF-TRANS-FC (H-M-T-FC) Algorithm

Debashish Rout[1]
Department of Computer Science and Engineering, VSSUT, Burla, India1
debashish.rout1@gmail.com

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 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.

1.6 Our Contribution

In this paper, we have proposed a Hybridized algorithm, which we popularly call as H-M-T-FC.We have also performed empirical study and comparative performance analysis of H-M-T-FC with MTF,TRANS and FC using three data sets such as Calgary Corpus, Canterbury Corpus. Our experimental results show that H-M-T-FC 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 1H-M-T-FC 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. H-M-T-FC Algorithm

In this section, we have developed a hybridized algorithm by combining three basic List Accessing Algorithms. Then total access cost of H M-T-FC is calculated by using Calgary Corpus dataset.

3. Our proposed Algorithms

3.1 Concept and Ideas

H M-T-FC Algorithm: *Upon an access of item x in the list, generate a random number for the item x; if the random number is between 0-0.33 apply MTF, if 0.33-0.66 applies TRANS, if 0.66-1 applies FC.*

3.2 Pseudo Code

```
Inputs:
l: size of the List L
n: size of the request sequence $\sigma$
Outputs:
$C_{H\ M-T-FC:}$ Cost of H M-T-FC Algorithm
Notations:
$P_{i:-}$Position of $i^{th}$ item in the list, $1 \le i \le l$
$\sigma_j$:-$j^{th}$ scanned item in the request sequence, $1 \le j \le n$
C ($\sigma_j$): Access cost $\sigma_j$ in the list L
R($\sigma_j$):-Random number of $\sigma_j$ in between o and 1
Algorithms
Initialize $C_{H\ M-T-FC}=0$;
 for j=1 to n
  {
   read the request $\sigma_j$ in the $\sigma$;
   Scan $\sigma_{j\ in}$ the L;
     x=$\sigma_{j;}$
   Let $P_i$ be the position of x in L
     C(x) =$P_{i;}$
    $C_{H\ M-T-FC}=C_{H\ M-T-FC}+C(x)$;
   if R(x) < 0.33
            MTF is applied.
   else if R(x) >0.66
            TRANS is applied
   else
            FC is applied
      }
```

3.3 Illustration of HM-T-FC algorithm

The hybrid algorithm uses the concept of probability to reduce the total access cost. Let the list is 123 and request sequence is 112323. Whenever we access 1, cost is 1 and then a random number is generated for the element 1, i.e. 0.2.As the random number less than 0.33, then MTF algorithm is applied here. Then the list configuration changed according to the MTF algorithm i.e. discussed earlier. Likewise all the items from the request sequence are served and the total access cost is calculated.

E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

4. Experimental Analysis

4.1 Experimental Setup

The proposed algorithm H-M-T-FC 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 MTF, TRANS, FC and H-M-T-FC 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, TRANS, FC and H-M-T-FC are computed and recorded for each file as shown in table.

4.4 Experimental Results

For our experiments we have considered bytes 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.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_{TRANS}	C_{FC}	$C_{HM\text{-}T\text{-}FC}$	Max No of Iteration
book1	81	768771	9770030	7341592	20816076	9402428	10
book2	96	610856	8053681	6615032	14173760	7996905	10
bib	81	111261	2197756	1666207	1980467	2185119	10
progp	89	49379	716228	623538	794317	716050	10
progl	87	71646	871903	741409	1036350	870006	10
paper4	80	13286	178311	151385	304101	174959	10
paper6	80	13286	178311	151385	304101	176125	10
paper2	91	82199	1063525	830825	1988314	1030870	10
paper3	84	46526	614682	492691	1092907	608543	10
pic	513216	1604276	1401356	1980293	1605197	513216	10

Table 4.1.1 Access cost incurred by MTF, TRANS, FC and HM-T-FC for Calgary Corpus with Byte Parsing

Dataset	|LS|	|RS|	C_{MTF}	C_{TRANS}	C_{FC}	$C_{HM\text{-}T\text{-}FC}$	Max No of Iteration
alice29	74	152089	2025917	1517934	3566682	1950375	10
asyoulik	68	125179	1908414	1433172	2020409	1843168	10
lcet10	84	426754	5570790	4284198	8677053	5415092	10
plrabn12	81	481861	6473411	4648224	7094677	6076150	10
ptt5	159	513216	1609438	1405539	2046355	1607827	10
grammar.lsp	76	3721	46330	42537	66271	44797	10
kennedy.xls	256	1029744	21557619	17987714	24215710	14885712	10

Table 4.1.2 Access cost incurred by MTF, TRANS, FC and HM-T-FC for Canterbury Corpus with Byte Parsing

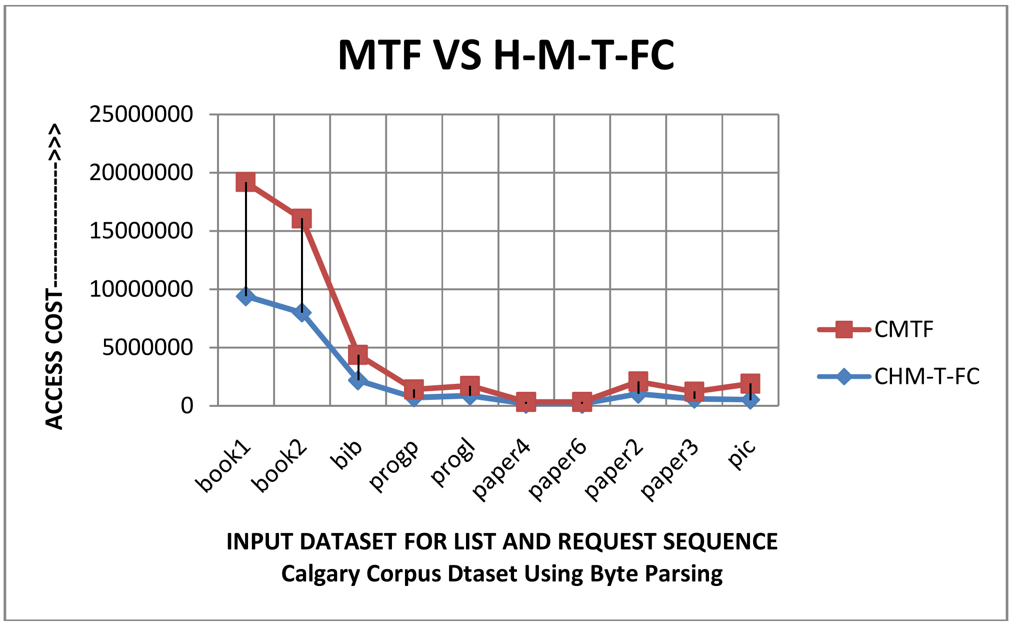

Fig:-4.1.1 Cost incurred by MTF and HM-T-FC against for Calgary Corpus

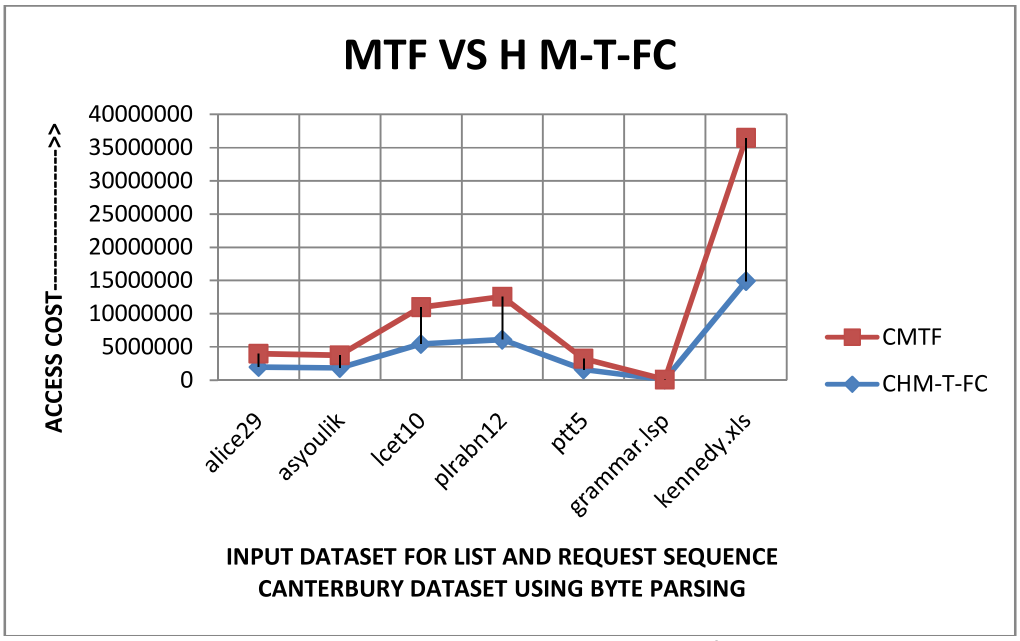

Fig: - 4.1.2 Cost incurred by MTF and HM-T-FC against for Calgary Corpus

5. Conclusion

List Accessing Algorithm take two parameter as input i.e. List and Request Sequence. we have proposed a Hybridized algorithm, which we popularly call as H-M-T-PCM.We have also performed empirical study and comparative performance analysis of H-M-T-FC with MTF,TRANS and FC using three data sets such as Calgary Corpus, Canterbury Corpus. Our experimental results show that H-M-T-FC outperforms for all request sequences for the two datasets, Calgary Corpus and Canterbury Corpus.

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.