Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 125 (2018) 762–769

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Conference on Smart Computing and Communications,ICSCC- 2017,
7-8 December 2017, Kurukshetra, India

New Results on Competitive Analysis of Move To Middle(MTM) List Update Algorithm using Doubly Linked List

Mohanty Rakesh, Singh Kumar Rakesh

[a]*Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Burla, Sambalpur, Odisha, India - 768018*
rakesh.iitmphd@gmail.com, kumarrakeshsingh2003@gmail.com,

Abstract

On-line algorithm is an emerging area of research since last five decades with immense theoretical and practical significance. Here the sequence of inputs is received and processed by the algorithm one by one in order. At any instant of time, the algorithm has the knowledge of past and present inputs without knowledge of future inputs. Competitive analysis is a standard performance measure for on-line algorithms in which the performance of an on-line algorithm is compared with that of an optimum offline algorithm. List update problem is a well studied research problem in the area of on-line algorithms, which finds applications in data-compression, dictionary maintenance, collision resolution in hash table, symbol table organization in compiler and computing convex hull in computational geometry. From the literature it is known that Move To Front(MTF) is the best performing deterministic list update algorithm using singly linked list in all practical applications. In this paper, we study and analyse the performance of a deterministic on-line list update algorithm known as Move To Middle(MTM) using doubly linked list. We make a first attempt to find the competitive ratio of MTM algorithm, which was only experimentally studied in the literature. In our study by considering doubly linked list as the data structure and a new variant of standard full cost model, we obtain new competitive analysis result and prove that MTM is 2-competitive. From our competitive analysis result, we show that MTM outperforms MTF.

Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications

Keywords: on-line algorithm; competitive analysis; list update problem; doubly linked list.

* Corresponding author. Mob.: +917008585072
E-mail address: rakesh.iitmphd@gmail.com

1877-0509 © 2018 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications
10.1016/j.procs.2017.12.098

1. Introduction

1.1. On-line Algorithms

In traditional algorithm, the whole set of inputs is available at the beginning and algorithm processes the whole set of inputs as a batch with the knowledge of complete information. Such algorithms are known as *off-line* algorithms. However in practical applications and real life scenarios, only the current and past inputs are known to the algorithm at any instant of time. The algorithm processes the inputs sequentially one by one in an interactive manner without the knowledge of the future inputs. Such algorithms are known as *on-line* algorithms. The on-line algorithms are mainly designed for interactive systems [1] and can produce output without the knowledge of entire input sequence. On-line algorithms are extensively applicable in various domains of computer science such as paging in virtual memory system, routing in communication networks, material task systems, load balancing, financial portfolio selection and self organizing data structures [2].

Formally we can define an on-line algorithm as follows. Let *ALG* be an on-line algorithm and let $\sigma = < \sigma_1, \sigma_2, \sigma_3, ..., \sigma_m >$ be an input sequence of finite length *m*. At a particular time *t* the input $\sigma_{t'}$ is not known to the algorithm *ALG*, where $t' > t$.

1.2. Competitive Analysis

Competitive analysis has gained the attention of researchers, since the introduction of seminal paper by Sleator and Tarjan in 1985 [3]. It is a standard performance measure for on-line algorithms. Here the performance of an on-line algorithm is measured by comparing its performance with that of an optimum off-line algorithm. The *optimum off-line* algorithm is one which incurs the minimum cost among all the off-line algorithms. Let $ALG(\sigma)$ and $OPT(\sigma)$ be the cost of the on-line algorithm *ALG* and optimum off-line algorithm *OPT* respectively on input sequence σ. Algorithm *ALG* is called *c-competitive*, if

$$ALG(\sigma) \leq c.OPT(\sigma) + k$$

Where c is a positive constant, $c \geq 1$ and k is an additive constant, $k \geq 0$. Here c is called the *competitive ratio*. *ALG* is called *strictly c-competitive*, when $k = 0$. The minimum value of the competitive ratio is always 1. When $c = 1$, on-line algorithm behaves exactly same as optimum off-line algorithm. Our objective is to design *competitive on-line algorithms*, where the value of c is bounded by a constant[2].

1.3. List Update Problem

List update problem [2] is a well studied cornerstone problem in the area of on-line algorithms and competitive analysis with immense practical and theoretical significance. In this problem we are given a list of distinct, unsorted items and a sequence of requests as inputs, where a request is one of the three operations such as access, insert or delete on an item of the list. As insert and delete operation involve access, we consider only access operation to simplify the analysis. When a requested item from the request sequence is accessed in the list, the request is served by incurring some access cost based on a cost model. We can reorganize the list at any time, while serving a request sequence on the list by incurring some reorganization cost. Our objective is to minimize the total cost of access and reorganization, while a request sequence is served on the list. The input list can be represented as a data structure such as a singly linked list or a doubly linked list or a tree. List update problem can be static or dynamic. In case of static only access operation is considered as request and in dynamic all the three operations such as access, insert and delete are considered. In our work we consider static list update problem with doubly linked list as the data structure

1.4. Practical Motivation

Applications of list update problem using singly linked list are data-compression, dictionary maintenance, collision resolution in hash table, symbol table organization in compiler, computing convex hull in computation geometry, arranging IP addresses in URL router routing table [6] and arranging records in cylinders of hard disk[8]. List update