# On Utilizing an Enhanced Object Partitioning Scheme to Optimize Self-Organizing Lists-on-Lists

O. Ekaba Bisong*, B. John Oommen†

**Abstract**

With the advent of "Big Data" as a field, in and of itself, there are at least three fundamentally new questions that have emerged, namely the Artificially Intelligence (AI)-based algorithms required, the hardware to process the data, and the methods to store and access the data efficiently. This paper[1] presents some novel schemes for the last of the three areas. There have been thousands of papers written regarding the algorithms themselves, and the hardware vendors are scrambling for the market share. However, the question of how to store, manage and access data, which has been central to the field of Computer Science, is even more pertinent in these days when megabytes of data are being generated every second. This paper considers the problem of minimizing the cost of retrieval using the most fundamental data structure, i.e., a Singly-Linked List (SLL). We consider a SLL in which the elements are accessed by a *Non-stationary* Environment (NSE) exhibiting the so-called "Locality of Reference". We propose a solution to the problem by designing an "Adaptive" Data Structure (ADS), which is created by utilizing a composite of hierarchical data "sub"-structures to constitute the overall data structure[2]. In this paper, we design hierarchical Lists-on-Lists (LOLs) by assembling a SLL into a hierarchical scheme that results in SLLs on SLLs (SLLs-on-SLLs) comprising of an outer-list and sublist contexts. The goal is that elements that are more likely to be accessed together are grouped within the same sub-context, while the sublists themselves are moved "en masse" towards the head of the list-context to minimize the overall access cost. This move is carried out by employing the "de-facto" list re-organization schemes, i.e., the Move-To-Front (MTF) and Transposition (TR) rules. To achieve the clustering of elements within the sublists, we invoke the Object Migration Automaton (OMA) family of reinforcement schemes from the theory of Learning Automata (LA). They capture the probabilistic dependence of the elements in the data structure, receiving query accesses from the Environment. We show that SLLs-on-SLLs augmented with the Enhanced OMA (EOMA) minimizes the retrieval cost for elements in NSEs, and are superior to the stand-alone MTF and TR schemes, *and also superior* to the OMA-augmented SLLs-on-SLLs operating in such Environments.

*This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. e-mail address: ekaba.bisong@carleton.ca.

†*Chancellor's Professor*; *Life Fellow: IEEE* and *Fellow: IAPR*. The author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. e-mail address: oommen@scs.carleton.ca. This author also holds an *Adjunct Professorship* with the Department of Information and Communication Technology, University of Agder, Grimstad, Norway. A preliminary version of this paper was presented at AIAI'19, the 2019 International Conference on Artificial Intelligence Applications and Innovations, in Crete, Greece, in May 2019.

[1]The work of the second author was partially supported by NSERC, the Natural Sciences and Engineering Council of Canada. We are very grateful for the feedback from the anonymous Referees of the *original* submission. Their input significantly improved the quality of this final version.

[2]In this paper, the primitive data structure is the SLL. However, these concepts can be extended to more complicated data structures.

**Keywords:** *Learning Automata, "Adaptive" Data Structures, Hierarchical Singly-linked Lists, Object Migration Automaton.*

# 1 Introduction

The goal of this research endeavor is to push the frontier of computational efficiency with regards to optimizing the speed of retrieving data from its data-structure. By considering the state-of-the-art, we address this issue by designing an *Adaptive* Data-Structure (ADS) that uses reinforcement learning schemes and their associated re-organization rules to update itself, as it encounters query accesses from the Environment of interaction. The consequence of this is the subsequent minimization of the resultant query access costs.

The paradigm that we advocate is quite novel, especially in these days when megabytes of data are being generated every second. Since the data itself can be large, we propose to split the data into smaller "chunks" of data that are most-likely accessed together. The jointly-accessed pieces are learned by invoking a Reinforcement Learning (RL)-based AI scheme. Now, when a data element is accessed, we propose that the entire data is reorganized in a multi-step way. First of all, the accessed element is adaptively moved towards the head of *its* data structure, i.e., the local "chunk". This "chunk" is then moved towards the head of the overall data, and this, too, is achieved in an adaptive manner. In this paper, we propose that the structures are Singly-Linked Lists (SLLs), which are, themselves, stored using SLLs whose individual components constitute the above-mentioned "chunks". The consequence of such a paradigm is that the query accesses are significantly enhanced, and simultaneously, the data elements which "should belong together" adaptively find their optimal locations.

As mentioned above, in this paper, the primitive data structure is the SLL. However, these concepts can be extended to more complicated structures, and so the avenues for further research are unbounded.

**The primitive sublists**: Of course, the immediate question is to understand what would constitute the elements of the primitive sublists. Our position is that they can be entirely arbitrary. They can be single fields of records, records themselves, pointers to images, pointers to albums of images, pointers to books, or even pointers to collections of books. Thus, for example, whenever a user accesses a book, it is moved towards the head of its sublist which would consist of a list of books (or pointers to the books), and the sublist itself would be moved "en masse" towards the head of the overall data structure. In the interest of simplicity, we shall refer to these elements of the sublists, in an abstract manner, as "objects". Thus, the model that we propose here can be made applicable for applications that involve "large data".

**The types of Environments**: Addressing now our specific slant, to render the problem realistic, the Environments under consideration in this work are considered to be time-varying, i.e., they are *Non-stationary* Environments (NSEs), where the elements' access probabilities change with time. As in the real world, we do not merely consider the case when these accesses are independent, because that would trivialize the problem. Instead, we assume that these Environments exhibit a particular dependency property called "Locality of Reference" where the events are probabilistically dependent on one another. In this work, we consider two such Environments, namely, the Periodic Switching Environment (PSE) and the Markovian Switching Environment (MSE).

To further illustrate the concept of a NSE, consider the volume of cars plying a major highway connecting two towns. At certain times or peak periods, the number of cars increases, and the highway becomes congested. However, at other periods, the traffic is more moderate. Also consider, as an example, an airport security checkpoint. At certain times of the day, large numbers of passengers will have to be processed for boarding, while at other off-peak periods, the security lanes are mostly idle. The phenomenon that the probabilities associated with the Environment change with respect to time is central to NSEs.

**The concept of Lists-on-Lists**: Expanding briefly on what we mentioned above, the approach we adopt in designing these "Adaptive" Data Structures (ADSs) is to set up a hierarchy of data "sub"-structures. In this research, we employ hierarchical Lists-on-Lists (LOL) data-structures pioneered by Amer and Oommen[3] in [2] for Singly-Linked Lists (SLLs) on Singly-Linked Lists. The LOL concept consists of an outer-list containing sublists. The elements of the outer-list and the containing sublists are called the list-context and sub-context, respectively. In this framework, elements that are more likely to be accessed together are grouped within the same sub-context, while the sublists are moved "en masse" towards the head of the list-context by following a re-organization rule.

**Capturing the probabilistic dependence**: To capture the probabilistic dependence of the elements in the data structure, based on the query accesses from the Environment, we employ a set of RL schemes derived from the theory of Learning Automata (LA). These RL schemes are variants of the so-called "Object Migration Automaton" (OMA).

**Crystallizing the prior art**: The pioneering work of Amer and Oommen in [2] utilized the OMA algorithm to capture the probabilistic dependence of the Environment's queries. The introduction of the OMA mitigated the static ordering of the sublists so that the elements can move freely from one sublist partition to another as the OMA learns the optimal sublist grouping. The addition of the OMA to the primitive hierarchical schemes resulted in the MTF-MTF-OMA, and TR-MTF-OMA, where the third component in the triple is LA used.**These issues will be clarified in greater detail in the body of the paper.**

Unfortunately, the OMA algorithm used in the literature [2] suffers from a deadlock[4] impediment that prevents it from converging to its optimal grouping. This is because the accessed element can be swapped from one sublist to another and then back to the original sublist. This deadlock phenomenon was mitigated by the Enhanced Object Migration Automaton (EOMA) in [13]. The EOMA forbids such "false alarm" swaps of elements between sublists and also avoids pointless swaps between the various sublists themselves. Moreover, the EOMA acknowledges that the sublist has converged when the elements are within a few of the most internal states. By this, it is sure about the identity of the elements that should constitute a sublist. This work augments the hierarchical SLLs-on-SLLs schemes with the EOMA. The design in this work yields the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and the TR-TR-EOMA schemes.

**Importance of the Problem in "Big Data"**: Given the increasing volume, velocity and variety of "Big Data", it is critical to devise efficient means for manipulating stored datasets. These big datasets may be stored in both structured and unstructured variants, and can quickly grow into magnitudes of gigabytes and terabytes. Examples of big data are sensor data streaming from IoT devices, data from genome se-

[3] Although this work, published in [2], set out some guidelines, as explained in the body of this paper, the results of this paper are far superior to those presented in that work.

[4] Although this is referred to as a "deadlock" in the literature, it could probably, be better termed as a "livelock".

quencing and large online retail businesses. Additionally, these datasets may be co-located or distributed geographically. These realities in the era of "Big Data" have made it necessary to devise new and improved data structures (or methods on existing data structurea) that can scale so as to improve the access cost of data retrieval, in the case of "Adaptive" data structures.

### 1.1 Contributions of this Paper

In summary, the novel contributions of this paper include:

- The design and implementation of the EOMA-enhanced SLLs-on-SLLs;
- The inclusion of the MTF-TR, and TR-TR enhanced hierarchical schemes as part of the SLLs-on-SLLs;
- Demonstrating the superiority of the EOMA-augmented hierarchical schemes to the MTF and TR rules when the outer-list context is the MTF;
- Demonstrating the superiority of the EOMA-augmented hierarchical schemes to the original OMA-augmented schemes that pioneered such LOL approaches, for "Periodic" and "UnPeriodic" versions;
- Showing that as the periodicity $T$ increases in the PSE, the asymptotic cost is further minimized.

### 1.2 Outline of this Paper

Section 1 makes a case for minimizing retrieval costs in NSEs. Section 2 surveys[5] the theory of LA, which forms the framework for the EOMA used in learning the true/best partition of objects into groups. The section addresses the concept of "Locality of Reference" in NSEs and outlays the models of dependence used in this work to simulate state probabilities. Section 4 discusses the "de-facto" MTF and TR adaptive list organizing schemes for NSEs, and why they constitute the primitive rules for the hierarchical LOL data-structures. Section 5 explains the rationale for using data "sub"-structures in designing the SLLs-on-SLLs giving rise to the MTF-MTF, MTF-TR, TR-MTF and TR-TR hierarchical schemes with static dependence capturing, making a case for an adaptive capturing mechanism. Section 6 explains the EOMA reinforcement algorithm and how it augments the Hierarchical SLLs-on-SLLs. Section 8 and 9 presents the results and discussions, and Section 11 concludes the paper.

## 2 Theoretical Background

### 2.1 The Field of Learning Automata

An Automaton, by definition, models an autonomous agent, whose *behavior* manifests as a consequence of the interplay between a sequence of stimuli from the Environment. The Automaton responds adaptively to the Environment and enforces the actions which fit the highest perceivable rewards from among a predetermined set of actions. Such an automaton is referred to as a *Learning Automaton* (LA) [9, 29].

[5]Due to space limitations, it is obvious that the background material will be surveyed very briefly. More details of the various concepts concerning LA can be found in [29], and the details of the applications of the OMA is found in the MCS thesis of the first author [9]. This thesis can be made available to the reader.

The theory of Learning Automata largely comprises of the *Fixed-Structure Stochastic Automata* (FSSA) and the *Variable-Structure Stochastic Automaton* (VSSA). In the FSSA, the states of the automata are characterized by Markov chains with time-invariant updates [29, 46] such as the Krinsky [19] and Krylov [20] automata. The VSSA offer more flexibility for modeling stochastic systems by incorporating functions that update the transition-action probabilities such as the Linear Reward-Penalty ($L_{R-P}$), Linear Reward-Inaction ($L_{R-I}$), Linear Inaction-Penalty ($L_{I-P}$), and the Linear Reward-$\epsilon$-Penalty ($L_{R-\epsilon P}$) schemes [22, 29]. These schemes have been extended to include nonlinear updating functions [22, 21, 29], with additional improvements in their speed of convergence being obtained when the updating functions are discretized versions of their continuous forms [23, 24, 31, 32, 35, 44, 51, 1].

LA have been used to provide solutions for a variety of applications including maintaining shortest path routing trees in stochastic networks [27], LA-based bus arbitration for shared-medium ATM switches [30] and other advances in telecommunications and networking [45] such as LA-based TDMA protocols for broadcast communication systems [39]. There have also been LA methods for call admission control [3], traffic control of high speed networks [4], LA-based method for sleep scheduling for partial coverage in wireless sensor networks [28], optimizing QoS routing in hierarchical ATM networks [48], analyzing stochastic properties of the random waypoint mobility model in wireless networks [8] and distributed LA for spectrum management in self-organized cognitive radio networks [10]. Besides, LA have found applications in global training of hidden Markov models [18], in developing LA-based adaptation of backpropagation algorithm parameters [26], and in analyzing spatial point patterns in GIS systems [6].

Of particular interest to this work is the FSSA, which form a sophisticated class of adaptive learning schemes for solving a variety of $NP$-hard optimization problems [11, 12]. It has been used to address the Object Partitioning Problem (OPP) by formulating the Object Migration Automaton (OMA) [37, 36], which is what the EOMA [13] enhances.

Oommen and Ma pioneered the OMA [37, 36] to solve a special case of the Object Partitioning Problem (OPP), namely the Equi-Partitioning Problem (EPP). The introduction of the OMA solution made real-life applications possible because the prior art [50] was an order of magnitude slower. The OMA resolved the EPP both efficiently and accurately, and it could thus be easily incorporated into *many* real-life application domains [9, 11, 12].

The OPP and its solutions have already found applications in several problem domains, including cloud computing, distributed databases, reputation systems, image retrieval, cryptanalysis and secure statistical databases, to mention just a few. We shall visit some of these individually.

- In the field of cloud computing, there is a need to distribute traffic across multiple Virtual Machines efficiently. An OMA-based solution, when applied to this problem, was superior to other solutions in the literature, leading to a 90% reduction in performance cost [17, 47].

- In cryptanalysis, the OMA has been used to solve the substitution-cipher model, which involves resolving the cipher using only the plaintext and the corresponding ciphertext [38].

- A distributed database system is a conceptual graph-like connection of distinct databases that may or may not be in physical proximity. Such a database configuration has to resolve the question of data

fragmentation and the cost of query access, which is $NP$-hard. The OMA outperforms many standard approaches for this problem [25].

- In reputation systems that take advantage of user feedback to provide recommendations, an OMA-based solution has been used to resolve the issue of "ballot stuffing" and "bad-mouthing" to boost the *trustworthiness* of the corresponding underlying reputation system [49].
- In retrieving similar images from databases with sub-directories, the retrieval and examination problem deals with grouping conceptually identical images into clusters. The OMA has been employed to solve this problem efficiently [33, 34].
- Finally, the OMA has been used for securing statistical databases by applying it to the Micro-Aggregation Problem (MAP), which involves assigning groups of individual records in a microdata file into mutually exclusive and exhaustive categories. The MAP is an $NP$-hard problem, and an OMA-based solution was shown to be superior to the state-of-the-art approaches to this problem [11, 12].

## 2.2 The OMA

In the partitioning problem, the underlying distribution of the objects among the classes is unknown to the OMA, and its goal is to migrate the objects between its classes, using the incoming queries specified by an Environment, denoted as $\mathbb{E}$. This should be done in such a way that the partitioning error is minimized as the queries are encountered. Such an Environment and its associated query generating system can be characterized by three main parameters, namely, the number of objects, specified by $W$, the number of groups or partitions, specified by $R$, and a quantity '$p$' , which is the probability specifying the certainty by which $\mathbb{E}$ pairs the elements in the query.

In our model, every query presented to the OMA by $\mathbb{E}$ consists of two objects, and this can be easily generalized for queries of larger sizes. Consider the case in which we have $3$ classes with $3$ objects, i.e., a system which has a total of 9 objects. This is depicted in Figure 1. $\mathbb{E}$ randomly selects an initial class with probability $\frac{1}{R}$, and it then chooses the first object in the query from it, say, $q_1$. The second element of the pair, $q_2$, is then chosen with the probability $p$ from the same class, and with the probability $(1-p)$ from one of the other classes uniformly, each of them being chosen with the probability of $\frac{1}{R-1}$.

In Figure 1, the three classes are named $G_1, G_2$ and $G_3$, and the objects inside them are represented by integers in $\{1, \cdots, 9\}$. The original distribution of the objects between the classes is shown in Figure 1, at the extreme left. This is the true unknown state of nature, i.e., $\Omega^*$. The OMA is initialized in a purely random manner by the numbers within the range. This step is depicted at the right of Figure 1, and $\Omega^0$ indicates the initial state of the OMA. At every iteration, a pair given by $\mathbb{E}$ is processed by the OMA, and it performs a learning step towards its convergence. The goal of the algorithm is for it to converge to a configuration, say $\Omega^+$. In an optimal setting, we would hope that $\Omega^+$ is identical to $\Omega^*$.

The OMA is a *Fixed Structure Stochastic Automata (FSSA)* designed to solve the EPP. It operates with so-called "Abstract" objects, $\{O_i\}$, that are used to represent the real-life physical objects, $\{A_i\}$. Thus, if the physical objects are each albums of images, the abstract objects are but pointers to these albums.

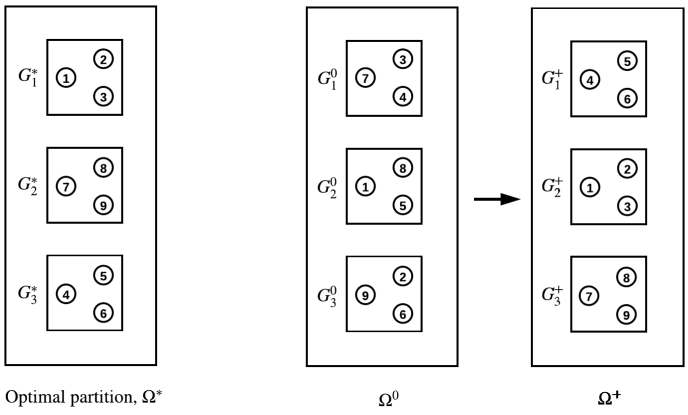



Figure 1: A figure describing the grouping of the objects, and the goal of a partitioning algorithm.

The OMA is defined as a quintuple with $R$ actions, each of which represents a specific class, and for every action, there exists a fixed number of states, $N$. Every abstract object from the set $\mathcal{O}$ resides in a state identified by a state number, and it can move from one state to another, or migrate from one group[6] to another. Thus, if the abstract object $O_i$ is located in state $\xi_i$ belonging to a specific group (action/class) $\alpha_k$, we say that $O_i$ is assigned to class $k$.

If two objects $O_i$ and $O_j$ happen to be in the same class and the OMA receives a query $\langle A_i, A_j \rangle$, they will be jointly rewarded by the Environment. Otherwise, they will be penalized. Our task is to formalize the movements of the $\{O_i\}$ on reward and penalty.

For every action $\alpha_k$, there is a set of states $\{\phi_{k1}, \cdots, \phi_{kN}\}$, where $N$ is the fixed depth of the memory, and where $1 \leq k \leq R$ represents the number of desired classes. We also assume that $\phi_{k1}$ is the most internal state and that $\phi_{kN}$ is the boundary state for the corresponding action. The reward and penalty responses are schematically shown in Figure 2, and defined as follows:

- **Reward:** Given a pair of physical objects presented as a query $\langle A_i, A_j \rangle$, if both $O_i$,and $O_j$ happen to be in the same class $\alpha_k$, the reward scenario is enforced, and they are both moved one step toward the actions's most internal state[7]. This is depicted in Figure 2 **(a)**.

- **Penalty:** If, however, they are in different classes, $\alpha_k$ and $\alpha_m$, (i.e., $O_i$, is in state $\xi_i$ where $\xi_i \in \{\phi_{k1}, \cdots, \phi_{kN}\}$ and $O_j$, is in state $\xi_j$ where $\xi_j \in \{\phi_{m1}, \cdots, \phi_{mN}\}$) they are moved away from $\phi_{k1}$ and $\phi_{m1}$ as follows (depicted in Figures 2 **(b)-(d)** respectively:

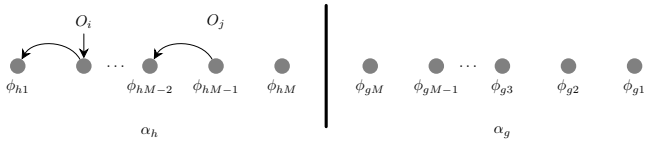
  1. If $\xi_i \neq \phi_{kN}$ and $\xi_j \neq \phi_{mN}$, then we move $O_i$ and $O_j$ one state toward $\phi_{kN}$ and $\phi_{mN}$, respectively.
  2. If $\xi_i = \phi_{kN}$ or $\xi_j = \phi_{mN}$ but not both (i.e., only one of these abstract objects is in the boundry state), the object which is not in the boundary state, say $O_i$, is moved towards *its* boundary state.

[6] As in the field of LA, we use the terms "action", "class" and "group" synonymously.

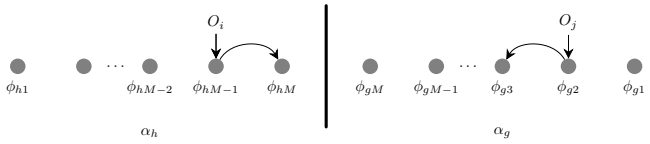
[7] The formal algorithm describing the LA is given in [9], and is omitted here in the interest of space, and also because we present, its enhanced version, the EOMA, with all its details. The algorithm invokes the procedures "*ProcessReward*" and "*ProcessPenalty*" also given algorithmically in [9].

Simultaneously, the object that is in the boundary state, $O_j$, is moved to the boundary state of $O_j$. Since this reallocation will result in an excess of objects in $\alpha_k$, we choose one of the objects in $\alpha_k$ (which is not accessed) and move it to the boundary state of $\alpha_m$. In this case, we choose the object nearest to the boundary state of $\xi_i$.

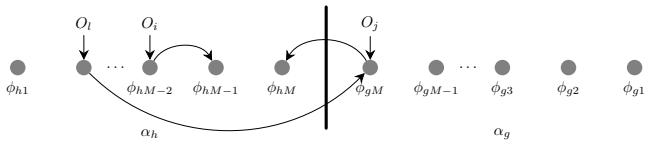
3. If $\xi_i = \phi_{kN}$ and $\xi_j = \phi_{mN}$ (both objects are in the boundary states), one object, say $O_i$, will be moved to the boundary state of $\alpha_m$. Since this reallocation will again result in an excess of objects in $\alpha_m$, we choose one of the objects in $\alpha_m$ (which is not accessed) and move it to the boundary state of $\alpha_k$. In this case, we choose the object nearest to the boundary state of $\xi_j$.



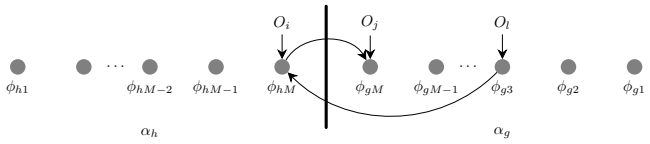

**(a)** On reward: Move the accessed abstract objects $\langle O_i, O_j \rangle$ towards the extreme states.




**(b)** On penalty: Move the accessed abstract objects $\langle O_i, O_j \rangle$ towards their boundary states.




**(c)** On penalty: Move the accessed abstract objects $\langle O_i, Oj \rangle$ to be in the same group. An extra object $O_l$ in the old group of $O_i$ is moved to the old group of $O_j$.




**(d)** On penalty: If both abstract objects $\langle O_i, O_j \rangle$ are in the boundary states, move one of them, say $O_i$, to the boundary state of the other group. An extra object $O_l$ in the group of $O_j$ is moved to the old group of $O_i$.

Figure 2: A figure displaying the schematic state transitions of the OMA.

# 3 Environments with Locality of Reference

Non-Stationary Environments (NSEs) deal primarily with learning in settings that change with time. Thus, a NSE has its penalty probabilities $\{c_i(t)\}$, in which $c_i(t)$ changes with time. In the context of an ADS, this variation affects the expected query cost because the Environment exhibits the so-called "Locality of Reference", or is characterized by dependent accesses. Locality of Reference occurs when there exists a probabilistic dependence between the consecutive queries [5]. Thus, there is a considerably small number of unrelated queries within a segment of the accesses.

Given a set of $n$ distinct elements, if we split them into $k$ disjoint and equal partitions with $m$ elements where $n = k.m$, the $k$ subsets can be considered to be local or "sub"-contexts. If the elements within a sub-context $k_i$ exhibit Locality of Reference, it implies that if an element from set $k_i$ is queried at time $t$, there exists a high likelihood that the next queried element will also arrive from the same set $k_i$. Thus, the Environment itself can be modeled to have a finite set of states $\{Q_i | 1 \leq i \leq k\}$, and the dependent model defines the transition from one Environmental state to another.

Learning schemes with fixed policies may become non-expedient over time, rendering them inadequate for such Environments. The goal is to have schemes that possess enough flexibility to choose actions that minimize the expected penalty. Two models of NSEs critical to this research are the Markovian Switching Environments (MSEs), and the Periodic Switching Environments (PSEs).

**Markovian Switching Environments (MSEs)**: Consider an Environment with 128 distinct records, that are divided into $k = 4$ subsets, with 32 elements in each subset, where the contiguous indices are considered to be in the same set (without loss of generality) for the sake of convenience. In such a case, the set of states $\{Q_1, Q_2, Q_3, Q_4\}$, could be $Q_1 = \{1 \ldots 32\}$, $Q_2 = \{33 \ldots 64\}$, $Q_3 = \{65 \ldots 96\}$, and $Q_4 = \{97 \ldots 128\}$. The Markovian Switching Environment (MSE) models each subset as the states of the Environment dictated by a Markov chain. If the probability of the Environment choosing a record from the current subset is 0.9, the probability of switching to another subset is equally divided among the other three subsets. After a query is generated from $Q_1$, the Environment remains in that state with probability $\alpha$ and moves to a different state with probability $\frac{1-\alpha}{k-1}$.

**Periodic Switching Environments (PSEs)**: The Periodic Switching Environment (PSE), on the other hand, changes the state of the Environment in a round-robin fashion, i.e., after every $T$ queries, the Environment changes state from $Q_i$ to $Q_{i+1 \bmod k}$. This implies that each set of $T$ consecutive queries belong to the same sub-context. Further, there are two variations that define the PSE model; the first is when the data structure is aware of the change of state in the query generator (*"Periodic"*), and the other is when the data structure is unaware of the state change (*"UnPeriodic"*). Understandably, the performance of the scheme is better when the ADS is aware of the Environment's state change.

## 3.1 Models of Dependence

The Environment generates queries according to a probability distribution. This work considers five different types of query distributions, namely, the Zipf, Eighty-Twenty, Lotka, Exponential and Linear distributions. For a given list of size $J$, divided into $k$ sublists, with each sublist containing $\frac{J}{k}$ elements, the probability distribution $\{s_i\}$ where $1 \leq i \leq m$ describes the query accesses for the elements in the subset $k$. Thus, the total probability mass for the accesses in each group is the same, and the distribution within each group has the specified distribution. The distributions for these generators are described below.

1. **The Zipf distribution**: The access probabilities for the Zipf query generator is given as: $s_i = \frac{1}{iH_m}$, for $1 \leq i \leq m$, where $H_m$ is the $m^{th}$ Harmonic number and defined as $H_m = \sum_{j=1}^{m}(\frac{1}{j})$. The Zipf distribution is the most commonly-used one for modelling real-life access probabilities.

2. **The 80-20 distribution**: The access probabilities for the 80-20 query generator is given as: $s_i =$

$\frac{1}{i^{(1-\theta)} H_m^{(1-\theta)}}$, for $1 \leq i \leq m$ and $\theta = \frac{\log 0.80}{\log 0.20} \approx 0.1386$, where $H_m^{(1-\theta)}$ is the $m^{th}$ Harmonic number of order $(1-\theta)$, and is given by $\sum_{j=1}^{m} (\frac{1}{j^{(1-\theta)}})$.

3. **The Lotka distribution**: The access probabilities for the Lotka query generator is given as: $s_i = \frac{1}{i^2 H_m^2}$, for $1 \leq i \leq m$, where $H_m^2$ is the $m^{th}$ harmonic number of order 2, and is given by $\sum_{j=1}^{m} (\frac{1}{j^2})$.

4. **The Exponential distribution**: The access probabilities for the Exponential query generator is given as: $s_i = \frac{1}{2^i K}$, for $1 \leq i \leq m$, where $K = \sum_{j=1}^{m} (\frac{1}{2^j})$.

5. **The Linear distribution**: The access probabilities for the Linear query generator is given as: $s_i = K(m - i + 1)$, for $1 \leq i \leq m$, where $K$ is determined as the constant which normalizes the $\{s_i\}$ to be a distribution.

A rationale for conducting the simulations with these query distributions is that, for the most part, they result in "L-shaped" graphs, which assign high probabilities to a small number of the sublist elements. This is true for the Exponential and Lotka distribution, and to an extent, for the Zipf distribution.

# 4 Adaptive Lists-on-Lists (LOL)

Self-organization is the ability for a list to re-order its constituent elements in response to queries from the underlying query system that serves as an Environment. The probability distribution of the query accesses is unknown to the list re-organization algorithm. The goal of this re-organization, among others, is to minimize the asymptotic cost or access-time of record retrieval.

The cost models employed in evaluating list access costs are the asymptotic cost, which is the ensemble mean of the final time-average cost after a convergence threshold, and the amortized cost, which is the mean overall query costs [7, 14, 42]. In studying ADSs, one assumes that the Environment will not request a record absent from the list and that each record is retrieved at least once [14].

The simplest and yet most prominent rules for Adaptive Lists are the Move-to-Front (MTF) and the Transposition rule (TR) schemes. The MTF update heuristic moves the queried element to the front of the list. In the TR, a queried record (if not at the front) is moved one position towards the front of the list.

For Environments with Locality of Reference, the MTF and TR have been shown to be superior to other deterministic schemes such as FC, MRI(0) and TS(0) [5]. Further, the time and space complexities involved in implementing other composite MTF and TR schemes (the details of which are omitted here) such as the MHD($k$) [40], the POS($k$) and the SWITCH($k$) [43] and other probabilistic approaches such as the $SPLIT$ algorithm [16], the JUMP [15], MTF2, Randomized MTF (RMTF), and the Randomized move ahead (RMHD) schemes [5] render most of them impractical for real-world settings.

# 5 Hierarchical Data "Sub"-structures

The novel idea that was proposed in [2], and that we advocate, is to combine the MTF and TR rules to take advantage of the quick updates of the MTF rule, and the asymptotically stable convergence of the TR rule, in designing the improved hierarchical strategies. The concept of a hierarchical data "sub"-structure involves

dividing a list of size $J$ into $k$ sublists. A re-organization strategy is then hierarchically applied to the list by first considering the elements within the sublist (also called the sub-context) and then operating over the sublists (or sub-contexts) themselves.

As mentioned earlier, the primitive re-organization strategies involved are the MTF and TR rules. When used in a hierarchical scheme, this yields the MTF-preceding-MTF, (MTF-MTF), MTF-preceding-TR, (MTF-TR), TR-preceding-MTF, (TR-MTF), and TR-preceding-TR, (TR-TR) schemes. For example, in the case of MTF-TR, the element within a sub-context is first moved to the front of the list, and then the sub-context is moved to the front of the entire list context.

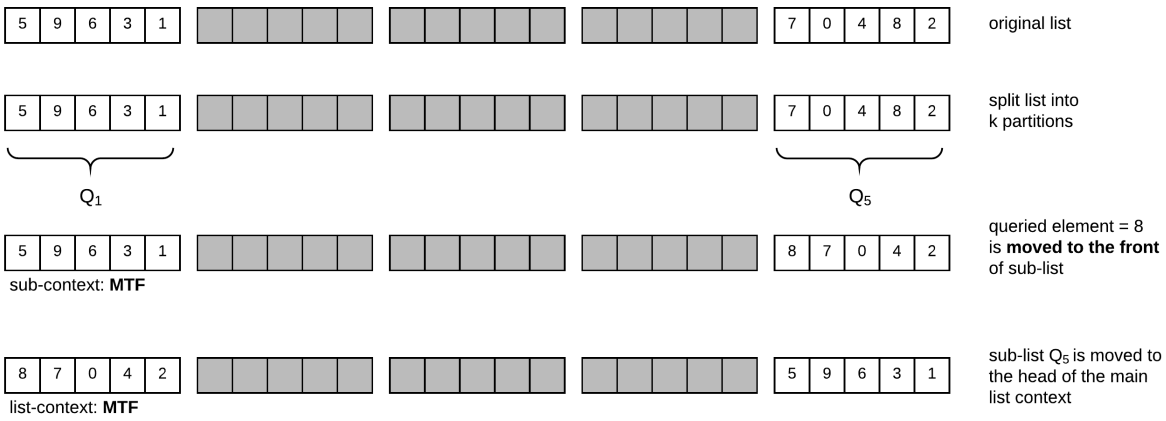



Figure 3: A diagrammatic description of the MTF-MTF Hierarchical scheme.

The hierarchical schemes on their own, however, perform worse than stand-alone schemes such as the MTF and TF in NSEs. The drawback is because the hierarchical schemes assume that the elements within a specific *a priori* sub-context have a probabilistic dependence. But this is often not the case as the elements in the list initially are ordered arbitrarily. To mitigate this shortcoming, we will later argue that we must design a mechanism to adaptively group the elements that have a probabilistic dependence within the same sub-context.

# 6 EOMA-Augmented Hierarchical SLLs-on-SLLs

The "Enhanced" OMA (EOMA) is an upgraded embodiment of the OMA algorithm proposed by the authors of [13] to mitigate the susceptibility of the OMA algorithm to a "deadlock situation" which prevents the algorithm from converging to the objects' optimal partitioning. This deadlock[8] condition, which is more accurately a "livelock", is exacerbated when the algorithm is interacting with a near-optimal Environment (e.g., when $p = 0.9$) by considerably slowing down the convergence rate even if the problem complexity is small.

[8]The details of the deadlock and of the method by which it is mitigated is quite involved. It is omitted here in the interest of space, but described in detail in [9]. However, the final resolution is briefly presented below.

The deadlock phenomenon occurs when there is a query pair $\langle O_i, O_j \rangle$ in a stream of query pairs belonging to different actions, $\alpha_m$ and $\alpha_k$. If one object is in the boundary state of its action, and the other is not, the query pairs are prevented from converging to their optimal ordering, and this can lead to an "infinite" loop scenario. To mitigate this, if there exists an object in the boundary state of the group containing $O_j$, the EOMA swaps $O_i$ with the object in this boundary state (Figure 4). Otherwise, the update is identical to the OMA. The algorithmic description for the EOMA is shown in Algorithms 1, 2 and 3.

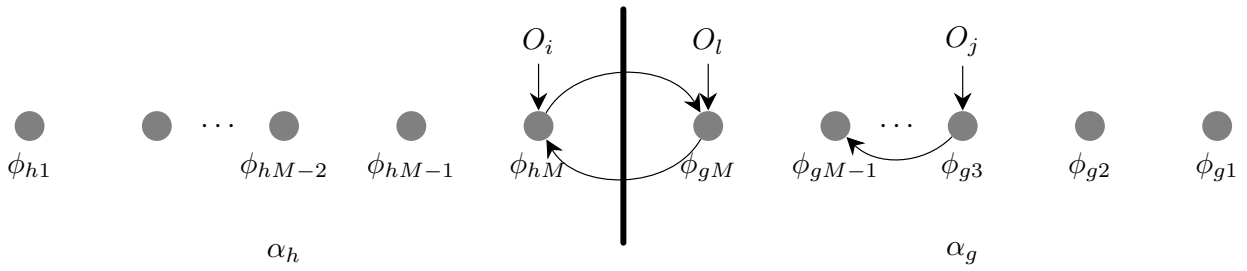



Figure 4: Resolving the deadlock scenario with the EOMA when only one object is in the boundary state.

**Algorithm 1** The EOMA Algorithm

**Input:**

- The abstract objects $\{O_1, \cdots, O_W\}$.
- The number of states $N$ per action.
- A stream of queries $\{\langle A_p, A_q \rangle\}$.

**Output:**

- A periodic clustering of the objects into R partitions.
- $\xi_i$ is the state of the abstract object $O_i$. It is an integer in the range $\{1, 2, \cdots, R \times N\}$, where, if $(k-1)N + 1 \leq \xi_i \leq kN$ then object $O_i$ is assigned to $\alpha_k$.

```
1:  begin
2:      Initialization of {ξ_p}
3:      for a sequence of T queries do
4:          Read query ⟨A_i, A_j⟩
5:          if ξ_i div N = ξ_j div N then                ▷ The partitioning is rewarded
6:              Call ProcessRewardEOMA({ξ_p}, A_i, A_j)
7:          else                                         ▷ The partitioning is penalized
8:              Call ProcessPenaltyEOMA({ξ_p}, A_i, A_j)
9:          end if
10:     end for
11:     Print out the partitions based on the states {ξ_i}
12: end
```

*Source: Excerpt from [41]*

The EOMA also redefines the concept of the convergence condition to reduce the algorithm's vulnerability to divergent queries. This modification designates the two-innermost states as the "final" states, as opposed to just the innermost state in the vanilla OMA. A marginally superior solution specifies a parameter $m$, to designate the $m$ innermost states of each action to be the convergence condition. More details on the EOMA are found in [13, 41].

The augmentation of the hierarchical SLLs based on the EOMA reinforcement scheme results in a new set of hierarchical strategies, namely, the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and the TR-

**Algorithm 2** $ProcessPenaltyEOMA(\{\xi_p\}, A_i, A_j)$

**Input:**

- The indices of the states, $\{\xi_p\}$.
- The query pair $\langle A_i, A_j \rangle$.

**Output:**

- The next states of the $O_i$'s.

**begin**
  **if** $\xi_i \bmod N \neq 0 \wedge \xi_j \bmod N \neq 0$ **then** ▷ Both are in internal states
    $\xi_i = \xi_i + 1$
    $\xi_j = \xi_j + 1$
  **else if** $\xi_i \bmod N \neq 0$ **then** ▷ $O_i$ is at internal state
    $\xi_i = \xi_i + 1$
    $temp = \xi_j$ ▷ Store the state of $O_j$
    $l =$ Index of the *unaccessed* object closest to the boundary state of $O_i$.
    $\xi_j = \xi_i$
    $\xi_l = temp$
  **else if** $\xi_j \bmod N \neq 0$ **then** ▷ $O_j$ is at internal state
    $\xi_j = \xi_j + 1$
    $temp = \xi_i$ ▷ Store the state of $O_i$
    $l =$ Index of the *unaccessed* object closest to the boundary state of $O_j$.
    $\xi_i = \xi_j$
    $\xi_l = temp$
  **else** ▷ Both are in boundary states
    $temp = \xi_i$ ▷ Store the state of $O_i$
    $\xi_i = \xi_j$ ▷ Move $O_i$ to the same group as $O_j$
    $l =$ index of an *unaccessed* object in group of $O_j$ closest to the boundary
    $\xi_i = \xi_j$
    $\xi_l = temp$ ▷ Move $O_l$ to the old state of $O_i$
  **end if**
**end**

*Source: Excerpt from [41]*

TR-EOMA.

## 6.1 Rationale and Time Complexity

Now that the framework has been laid, before we proceed, it is prudent to explain why[9] the proposed methodology works well, and to present some thoughts on the time efficiency of what we have proposed.

When we consider the relatively straightforward tasks of storing, accessing and processing data, the problem is very complex, especially in the context of the "Big Data" phenomenon. Consequently, since moving and reorganizing large amounts of data is a problem in itself, a common strategy would be to *partition* the data, and then to try to optimize that storage and retrieval of the partitions themselves. This is precisely our strategy. The question, then, is one of knowing how the partitioning can be achieved. We propose to do this using LA, and in particular, the OMA family of machines, as we have explained. Once the data has been partitioned, we now propose to move the individual chunks (i.e., the sub-partitions) of data within the overall data itself. One could do this by moving these chunks one step at a time or

[9] We are grateful to the Anonymous Referee who requested this.

**Algorithm 3** $ProcessRewardEOMA(\{\xi_p\}, A_i, A_j)$

**Input:**

- The indices of the states, $\{\xi_p\}$.
- The query pair $\langle A_i, A_j \rangle$.

**Output:**

- The next states of the $O_i$'s.

```
1: begin
2:     if ξ_i mod N ≠ 1 then                    ▷ Move O_i towards the internal state
3:         ξ_i = ξ_i − 1
4:     end if
5:     if ξ_j mod N ≠ 1 then                    ▷ Move O_j towards the internal state
6:         ξ_j = ξ_j − 1
7:     end if
8: end
```

*Source: Excerpt from [41]*

drastically towards the front (head) of the underlying data structure. One could also move elements of any sub-partition itself ahead of other elements within the specific sub-partition. Our proposed scheme does both of the operations using the primitive ADS list operations.

With regard to the time complexity, the time required to move the access data element within the partition is linear because the OMA primarily only comparing two elements at a time. However, in the case when the data element to be migrated is at the boundary state associated with a partition, we have to do some additional searching to only recognize the specific element closest to the boundary of the class to which it is being moved. This additional cost is minimal if the book-keeping is done efficiently.

## 6.2 MTF-MTF-EOMA

The MTF-preceding-MTF (MTF-MTF) incorporates the EOMA partitioning algorithm into the hierarchical scheme to yield the MTF-MTF-EOMA strategy. In this scheme, the groups resulting from the execution of the EOMA, as it learns the dependency model of the Environment, are used to adjust the memberships of the sublists of the SLL on SLL methods.

The ability of the sub-list to contain elements that are probabilistically dependent as per the Environment, breaks the static sub-list constraint observed in the MTF-MTF scheme when it is not enhanced with the EOMA-based reinforcement paradigm. Moreover, as alluded to earlier, this, in turn, leads to a superior performance to the standalone MTF and TR schemes in NSEs that possess a "Locality-of-Reference".

In the MTF-MTF-EOMA scheme, if the abstract objects $\langle O_i, O_j \rangle$ of the query pairs $\langle A_i, A_j \rangle$ are in the same group, it results in a reward to the EOMA. On reward, the current element $A_j$ is moved to the front of its sub-list, while the sub-list is, in turn, moved to the front of the list context. Also, if the query pairs are in different groups (or classes), it results in a penalty for the EOMA. On being penalized, if both abstract objects $\langle O_i, O_j \rangle$ are in their internal states, the current element $A_j$ is moved to the head of its sub-list context.

If one of the abstract objects $\langle O_i, O_j \rangle$ is in a boundary state, e.g., $O_i$, the EOMA ensures that both elements $\langle A_i, A_j \rangle$ are in the same sub-list by swapping the element $A_i$ with another element closest to the boundary state of the element $A_j$. This move, in particular, breaks the OMA's deadlock situation, which only swaps

the elements when both the abstract objects are in a boundary state. Finally, if both abstract objects $\langle O_i, O_j \rangle$ are in the boundary states of their respective groups (or classes), the elements $\langle A_i, A_j \rangle$ are swapped in-line with the corresponding EOMA's abstract objects $\langle O_i, O_j \rangle$ as seen in Algorithm 1.

The reader should observe that, throughout the iteration phase of the MTF-MTF-EOMA, the list structure of the MTF-MTF mirrors the classes/groups represented by the EOMA's abstract objects, $\mathcal{O}$. The algorithmic description of the MTF-MTF-EOMA is given in Algorithm 4.

**Algorithm 4** The MTF-MTF-EOMA Algorithm

**Input:**

- The abstract objects $\{O_1, \cdots, O_W\}$.
- The number of states $N$ per action.
- A stream of queries $\{\langle A_p, A_q \rangle\}$.
- The list-on-list data structure with equi-partitioned elements, $A$.
- $\xi$ is the states of the abstract objects $O$.

**Output:**

- The updated list using the MTF-MTF-EOMA rule.

**begin**
  Initialization of $\{\xi_p\}$
  **for** a sequence of $T$ queries **do**
    Read query $\langle A_i, A_j \rangle$
    **if** ProcessReward($\{\xi_p\}, A_i, A_j$) **then** ▷ The partitioning is rewarded
      Get sub-list partition, $k$ containing the query $A_j$
      Move element $A_j$ to the head of sub-list, $k$.
      Move sub-list $k$ to the head of the list, $A$.
    **else** ProcessPenalty($\{\xi_p\}, A_i, A_j$) ▷ The partitioning is penalized
      **if** Both are in internal states **then**
        Get sub-list $k$ containing the query $A_j$
        Move element $A_j$ to the head of sub-list, $k$.
      **else if** $O_i$ is at boundary state **then**
        Move element $A_i$ to be in same sublist with $A_j$.
        Move element $A_k$ closest to boundary state of $k_i$ to $k_j$.
      **else if** $O_j$ is at boundary state **then**
        Move element $A_j$ to be in same sublist with $A_i$.
        Move element $A_k$ closest to boundary state of $k_j$ to $k_i$.
      **else** Both are in boundary states
        Remove $A_j$ from sub-list $k_j$.
        Get sub-list $k_l$ containing the element $l$.
        Remove $l$ from sub-list $k_l$. ▷ Note that $k_l == k_i$
        Append element $l$ to sub-list $k_j$.
        Append element $A_j$ to sub-list $k_i$.
      **end if**
    **end if**
  **end for**
**end**

*Note:* l is the index of the unaccessed object closest to the boundary of group $O_j$.

## 6.3 MTF-TR-EOMA

The MTF-preceding-TR (MTF-TR) can also be enhanced with an EOMA partitioning phase to yield the MTF-TR-EOMA scheme.

In the MTF-TR-EOMA scheme, if the abstract objects $\langle O_i, O_j \rangle$ of the query pairs $\langle A_i, A_j \rangle$ are in the same

group, it results in a reward for the EOMA. On reward, the current query $A_j$ is moved to the front of its sub-list, $k$, while the sub-list, $k$ is swapped with its preceding sub-list $k-1$ in the list context. Also, as per the EOMA algorithm, if the query pairs are in different groups (or classes), it results in a penalty. On penalty, if the abstract objects $\langle O_i, O_j \rangle$ are in their internal states, the current query $A_j$ is moved to the head of its sub-list context. Whereas, if the abstract objects $\langle O_i, O_j \rangle$ are in the boundary states of their respective groups (or classes), the elements $\langle A_i, A_j \rangle$ are swapped in-line with the corresponding EOMA abstract objects $\langle O_i, O_j \rangle$ as seen in Algorithm 1.

Again, given $\langle O_i, O_j \rangle$, if one abstract object is on a boundary state of its class and the other is not, e.g., $O_i$, the EOMA ensures that both elements $\langle A_i, A_j \rangle$ are in the same sub-list by swapping the element $A_i$ with another element closest to the boundary state of the element $A_j$. At every time step, the list structure of the MTF-TR rule is made to reflect the groups of the EOMA's abstract objects, $\mathcal{O}$. The algorithmic description of the MTF-TR-EOMA is given in Algorithm 5[10].

### 6.4 TR-MTF-EOMA

The TR-preceding-MTF (TR-MTF) can be improved with an EOMA-based partitioning. This yields the TR-MTF-EOMA scheme.

As before, in the TR-MTF-EOMA scheme, if the abstract objects $\langle O_i, O_j \rangle$ of the query pairs $\langle A_i, A_j \rangle$ are in the same group, it results in a reward for the EOMA. On reward, the current query $A_j$ is swapped with its preceding element $A_{j-1}$ in its sub-list context $k$, while the sub-list, $k$ is moved to the front of the list context. On penalty, the analysis operations are done as in Section 6.2 and 6.3, and the detailed descriptions are omitted to avoid repetition. We note though that the list structure of the TR-MTF rule is made to contain the corresponding grouping information represented by the EOMA's abstract objects, $\mathcal{O}$. The algorithmic description of the TR-MTF-EOMA is given in Algorithm 6.

### 6.5 TR-TR-EOMA

The TR-preceding-TR (TR-TR) scheme can also be modified to incorporate the EOMA paradigm. The corresponding hierarchical scheme is referred to as the TR-TR-EOMA scheme.

In the TR-TR-EOMA scheme, if the abstract objects $\langle O_i, O_j \rangle$ of the query pairs $\langle A_i, A_j \rangle$ are in the same group, it results in a reward transmitted to the EOMA. On being rewarded, the current query $A_j$ is swapped with its preceding element $A_{j-1}$ in its sub-list context $k$, and the sub-list, $k$ is also swapped with its preceding sub-list $k-1$ in the list context. As before, if the query pairs are in different groups (or classes) as per the EOMA algorithm, it results in a penalty. The penalty schemes operate as in Sections 6.2, 6.3 and 6.4, and the actual lists are made to reflect the class groups dictated by the EOMA. The algorithmic description of the TR-TR-EOMA is given in Algorithm 7.

The following sections of this paper report, in detail, the performance of these schemes. Additional simulation results are reported in the Appendix, so that the flow of the paper is not compromised.

[10]The actual algorithms are given formally in each of these cases to ensure that the EOMA's migration and the sublists/list operations are explained accurately.

**Algorithm 5** The MTF-TR-EOMA Algorithm

**Input:**

- The abstract objects $\{O_1, \cdots, O_W\}$.
- The number of states $N$ per action.
- A stream of queries $\{\langle A_p, A_q \rangle\}$.
- The list-on-list data structure with equi-partitioned elements, $A$.
- $\xi$ is the states of the abstract objects $O$.

**Output:**

- The updated list using the MTF-TR-EOMA rule.

**begin**
  Initialization of $\{\xi_p\}$
  **for** a sequence of $T$ queries **do**
    Read query $\langle A_i, A_j \rangle$
    **if** ProcessReward($\{\xi_p\}, A_i, A_j$) **then** ▷ The partitioning is rewarded
      Get sub-list partition, $k$ containing the query $A_j$
      Move element $A_j$ to the head of sub-list, $k$.
      **if** index($k$) ! $= 0$ **then** ▷ If $k$ is not the list head
        Swap sub-list $k$ with its preceding sub-list, $k-1$.
      **end if**
    **else** ProcessPenalty($\{\xi_p\}, A_i, A_j$) ▷ The partitioning is penalized
      **if** Both are in internal states **then**
        Get sub-list $k$ containing the query $A_j$
        Move element $A_j$ to the head of sub-list, $k$.
      **else if** $O_i$ is at boundary state **then**
        Move element $A_i$ to be in same sublist with $A_j$.
        Move element $A_k$ closest to boundary state of $k_i$ to $k_j$.
      **else if** $O_j$ is at boundary state **then**
        Move element $A_j$ to be in same sublist with $A_i$.
        Move element $A_k$ closest to boundary state of $k_j$ to $k_i$.
      **else** Both are in boundary states
        Remove $A_j$ from sub-list $k_j$.
        Get sub-list $k_l$ containing the element $l$.
        Remove $l$ from sub-list $k_l$. ▷ Note that $k_l == k_i$
        Append element $l$ to sub-list $k_j$.
        Append element $A_j$ to sub-list $k_i$.
      **end if**
    **end if**
  **end for**
**end**

*Note:* l is the index of the unaccessed object closest to the boundary of group $O_j$.

**Algorithm 6** The TR-MTF-EOMA Algorithm

**Input:**

- The abstract objects $\{O_1, \cdots, O_W\}$.
- The number of states $N$ per action.
- A stream of queries $\{\langle A_p, A_q \rangle\}$.
- The list-on-list data structure with equi-partitioned elements, $A$.
- $\xi$ is the states of the abstract objects $O$.

**Output:**

- The updated list using the TR-MTF-EOMA rule.

**begin**
  Initialization of $\{\xi_p\}$
  **for** a sequence of $T$ queries **do**
    Read query $\langle A_i, A_j \rangle$
    **if** ProcessReward($\{\xi_p\}, A_i, A_j$) **then** ▷ The partitioning is rewarded
      Get sub-list partition, $k$ containing the query $A_j$
      **if** index($A_j$) ! $= 0$ **then** ▷ If $A_j$ is not the list head
        Swap element $A_j$ with its preceding element, $A_{j-1}$ in sub-list $k$.
      **end if**
      Move sub-list, $k$ to the head of the list, $A$.
    **else** ProcessPenalty($\{\xi_p\}, A_i, A_j$) ▷ The partitioning is penalized
      **if** Both are in internal states **then**
        Get sub-list partition, $k$ containing the query $A_j$
        **if** index($A_j$) ! $= 0$ **then** ▷ If $A_j$ is not the list head
          Swap element $A_j$ with its preceding element, $A_{j-1}$ in sub-list $k$..
        **end if**
      **else if** $O_i$ is at boundary state **then**
        Move element $A_i$ to be in same sublist with $A_j$.
        Move element $A_k$ closest to boundary state of $k_i$ to $k_j$.
      **else if** $O_j$ is at boundary state **then**
        Move element $A_j$ to be in same sublist with $A_i$.
        Move element $A_k$ closest to boundary state of $k_j$ to $k_i$.
      **else** Both are in boundary states
        Remove $A_j$ from sub-list $k_j$.
        Get sub-list $k_l$ containing the element $l$.
        Remove $l$ from sub-list $k_l$. ▷ Note that $k_l == k_i$
        Append element $l$ to sub-list $k_j$.
        Append element $A_j$ to sub-list $k_i$.
      **end if**
    **end if**
  **end for**
**end**

*Note:* l is the index of the unaccessed object closest to the boundary of group $O_j$.

**Algorithm 7** The TR-TR-EOMA Algorithm

**Input:**

- The abstract objects $\{O_1, \cdots, O_W\}$.
- The number of states $N$ per action.
- A stream of queries $\{\langle A_p, A_q \rangle\}$.
- The list-on-list data structure with equi-partitioned elements, $A$.
- $\xi$ is the states of the abstract objects $O$.

**Output:**

- The updated list using the TR-TR-EOMA rule.

**begin**
  Initialization of $\{\xi_p\}$
  **for** a sequence of $T$ queries **do**
    Read query $\langle A_i, A_j \rangle$
    **if** ProcessReward($\{\xi_p\}, A_i, A_j$) **then** ▷ The partitioning is rewarded
      Get sub-list partition, $k$ containing the query $A_j$
      **if** index($A_j$) ! $= 0$ **then** ▷ If $A_j$ is not the list head
        Swap element $A_j$ with its preceding element, $A_{j-1}$ in sub-list $k$.
      **end if**
      **if** index($k$) ! $= 0$ **then** ▷ If $k$ is not the list head
        Swap sub-list $k$ with its preceding sub-list, $k-1$.
      **end if**
    **else** ProcessPenalty($\{\xi_p\}, A_i, A_j$) ▷ The partitioning is penalized
      **if** Both are in internal states **then**
        Get sub-list partition, $k$ containing the query $A_j$
        **if** index($A_j$) ! $= 0$ **then** ▷ If $A_j$ is not the list head
          Swap element $A_j$ with its preceding element, $A_{j-1}$ in sub-list $k$.
        **end if**
      **else if** $O_i$ is at boundary state **then**
        Move element $A_i$ to be in same sublist with $A_j$.
        Move element $A_k$ closest to boundary state of $k_i$ to $k_j$.
      **else if** $O_j$ is at boundary state **then**
        Move element $A_j$ to be in same sublist with $A_i$.
        Move element $A_k$ closest to boundary state of $k_j$ to $k_i$.
      **else** Both are in boundary states
        Remove $A_j$ from sub-list $k_j$.
        Get sub-list $k_l$ containing the element $l$.
        Remove $l$ from sub-list $k_l$. ▷ Note that $k_l == k_i$
        Append element $l$ to sub-list $k_j$.
        Append element $A_j$ to sub-list $k_i$.
      **end if**
    **end if**
  **end for**
**end**

*Note:* l is the index of the unaccessed object closest to the boundary of group $O_j$.

# 7 Settings and Parameters for the Experiments

The experimental setup involved a list of size 128, split into $k$ sublists, where $k \in \{2, 4, 8, 16, 32, 64\}$. In the MSE, the probability of subsequent query accesses coming from the same sublist, $\alpha$, was set to $0.9$. At the same time, the PSE had the hyper-parameter for the number of queries to arrive from the query space before switching to another pattern, $T$ set to $T = 30$. For all the results reported in this section, the simulation setup involved an ensemble of 10 experiments, each consisting of 300,000 query accesses.

The results for the different distributions and models of non-stationarity are given below. A few of these tables are included in the body of the paper itself with their standard deviation and confidence interval estimates[11]. Besides, additional simulation results are reported in the Appendix, so that, as mentioned above, the flow and readability of the paper is not compromised. We should, however, add that we did not use any off-the-shelf simulator. Instead, we wrote the simulator with all its modules, aspects and metric evaluations.

# 8 Results and Discussions: Performance in MSEs

From the simulation results in Table 1, with $k = 2$, we observed that the hierarchical schemes with EOMA generally outperformed their stand-alone counterparts in both the asymptotic (top of the table) and amortized (bottom of the table) costs for all instances excepting the Lotka and Exponential distributions. The stand-alone MTF and TR schemes had a slightly superior performance to the EOMA-augmented hierarchical schemes in the Lotka distribution, and comparable results in the Exponential distribution.

**Comparisons for the Linear distribution**: To cite an example, for the Linear distribution in Table 1, when $k = 2$, the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA had asymptotic and amortized costs of (39.25, 39.27, 35.73, 35.77) and (39.36, 39.37, 36.22, 36.20) respectively compared to the stand-alone MTF and TR schemes which had asymptotic and amortized costs of (54.84, 48.29) and (54.87, 49.50) respectively. The reader can readily observe that the hierarchical results are superior to the stand-alone versions. The confidence interval and the standard deviation estimates gives further evidence to the statistical significance of the superiority of the EOMA-augmented hierarchical schemes.

**Comparisons for the Exponential distribution**: However, if we consider the Exponential distribution, we realized that the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA with asymptotic and amortized costs of (3.30, 3.80, 2.48, 2.96) and (3.33, 3.84, 2.70, 3.18) respectively had comparable performance to the corresponding stand-alone MTF and TR schemes, with asymptotic and amortized costs of (2.81, 2.54) and (2.83, 2.99). By observing the standard deviations (SD) and confidence intervals (CI) of the EOMA-augmented schemes and the stand-alone MTF and TR schemes, we deduce that at certain instances the results of the EOMA-augmented schemes may be superior to that of the stand-alone versions even in an Environment with an L-shaped distribution that traditionally favors the 'de-facto' MTF and TR adaptive schemes. As an example of this, in the asymptotic costs of the Exponential distribution, the SD and CI es-

[11] We are grateful to the Anonymous Referee who requested these metrics. This required, possibly, hundreds of hours of additional computational time. To satisfy this Referee, we re-did the entire sets of experiments to record the additional metrics explained here. These new results have certainly increased the quality of the paper significantly.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 45.85 | 0.28 | (45.71, 45.98) | 49.66 | 0.29 | (49.52, 49.8) | 24.17 | 0.31 | (24.02, 24.32) | 2.81 | 0.03 | (2.79, 2.83) | 54.84 | 0.29 | (54.7, 54.98) |
| TR | 37.54 | 0.23 | (37.44, 37.65) | 41.39 | 0.29 | (41.25, 41.53) | 19.11 | 0.26 | (18.99, 19.24) | 2.54 | 0.04 | (2.52, 2.56) | 48.29 | 0.28 | (48.16, 48.43) |
| MTF-MTF -EOMA | 35.13 | 0.25 | (35.01, 35.26) | 36.84 | 0.24 | (36.72, 36.95) | 27.2 | 0.43 | (27.0, 27.41) | 3.3 | 2.1 | (2.29, 4.31) | 39.25 | 0.26 | (39.12, 39.38) |
| MTF-TR -EOMA | 35.19 | 0.28 | (35.05, 35.32) | 36.91 | 0.37 | (36.74, 37.09) | 27.1 | 0.33 | (26.94, 27.26) | 3.8 | 2.96 | (2.38, 5.22) | 39.27 | 0.23 | (39.17, 39.38) |
| TR-MTF -EOMA | 31.6 | 0.34 | (31.34, 31.85) | 33.36 | 0.29 | (33.14, 33.59) | 24.45 | 0.25 | (24.26, 24.64) | 2.48 | 0.03 | (2.45, 2.5) | 35.73 | 0.19 | (35.58, 35.87) |
| TR-TR -EOMA | 31.6 | 0.29 | (31.46, 31.74) | 33.14 | 0.29 | (33.0, 33.28) | 24.69 | 0.64 | (24.39, 25.0) | 2.96 | 2.01 | (1.99, 3.92) | 35.77 | 0.28 | (35.64, 35.9) |
| MTF | 45.83 | 0.13 | (45.77, 45.9) | 49.6 | 0.17 | (49.52, 49.68) | 24.07 | 0.17 | (23.98, 24.15) | 2.83 | 0.01 | (2.82, 2.83) | 54.87 | 0.13 | (54.81, 54.93) |
| TR | 38.71 | 0.2 | (38.61, 38.81) | 42.7 | 0.26 | (42.57, 42.82) | 20.29 | 0.16 | (20.21, 20.37) | 2.99 | 0.1 | (2.95, 3.04) | 49.5 | 0.18 | (49.41, 49.59) |
| MTF-MTF -EOMA | 35.36 | 0.18 | (35.28, 35.45) | 36.97 | 0.17 | (36.89, 37.05) | 28.59 | 0.7 | (28.26, 28.93) | 3.33 | 2.14 | (2.3, 4.36) | 39.36 | 0.16 | (39.28, 39.44) |
| MTF-TR -EOMA | 35.41 | 0.2 | (35.31, 35.5) | 37.01 | 0.16 | (36.93, 37.09) | 29.04 | 1.53 | (28.31, 29.78) | 3.84 | 2.99 | (2.4, 5.27) | 39.37 | 0.13 | (39.31, 39.43) |
| TR-MTF -EOMA | 32.22 | 0.25 | (32.03, 32.4) | 33.89 | 0.3 | (33.66, 34.12) | 28.94 | 3.18 | (26.54, 31.33) | 2.7 | 0.02 | (2.69, 2.71) | 36.22 | 0.11 | (36.14, 36.31) |
| TR-TR -EOMA | 32.36 | 0.37 | (32.19, 32.54) | 33.75 | 0.21 | (33.65, 33.85) | 28.51 | 2.73 | (27.2, 29.83) | 3.18 | 2.11 | (2.17, 4.2) | 36.2 | 0.17 | (36.12, 36.28) |

Table 1: Experimental results displaying the average number of accesses for the hierarchical schemes that include the EOMA and the stand-alone schemes in MSEs, where $\alpha = 0.9$ and $k = 2$. For the hierarchical schemes, a list of size **128** was split into **2** sublists with **64** records each. The **top** portion of the table shows the asymptotic cost, while the amortized cost is at the **bottom**.

timates are tighter in the standalone schemes with (SD: 0.03, CI: [2.79, 2.83]) for the MTF and (SD: 0.04, CI: [2.52, 2.56]) for TR but wider in the EOMA-augmented schemes with (SD: 2.1, CI: [2.29, 4.31]) for MTF-MTF-EOMA, (SD: 2.96, CI: [2.38, 5.22]) for MTF-TR-EOMA, (SD: 0.03, CI: [2.45, 2.5]) for TR-MTF-EOMA and (SD: 2.01, CI: [1.99, 3.92]) for the TR-TR-EOMA scheme.

**Comparisons for the Lotka distribution**: It is noteworthy that the Lotka distribution with asymptotic and amortized costs for the MTF and TR rules given as (24.17, 19.11) and (24.07, 20.29) are superior to those for the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA which are (27.20, 27.10, 24.45, 24.69) and (28.59, 29.04, 28.94, 28.51). As observed, the MTF and TR rules are competitive in Environments with an L-shaped logarithmic curve such as the Exponential and Lotka distributions, because they assign higher probabilities to a small subset of the elements in the query system.

**Behavior as the sub-list sizes decreased**: Further, we considered the behavior of the EOMA-based hierarchical schemes (and the stand-alone schemes) as the number of elements within the sub-lists decreased due to an increasing number of list partitions. From Table 2, we observed that the MTF-MTF-EOMA and the TR-MTF-EOMA schemes yielded superior results compared to the stand-alone schemes for all distributions except the Exponential distribution. In the Exponential distribution, although the asymptotic and amortized costs for the MTF is slightly superior to the MTF-MTF-EOMA and the TR-MTF-EOMA schemes, these EOMA-augmented schemes are still significantly superior to the TR rule in this competitive distribution.

On the other hand, the MTF-TR-EOMA and the TR-TR-EOMA results were inferior when compared to the MTF rule. That being said, all the hierarchical schemes performed better than the TR scheme, which

performed poorly as the number of partitions increased in Environments with dependent queries. Interestingly, just as the TR scheme performed poorly when compared to the MTF scheme, the hierarchical schemes that invoked the TR rule as the outer-list context of the hierarchical formulation, also performed poorly when compared with their hierarchical counterparts that had the MTF rule as the outer-list context.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 20.76 | 0.41 | (20.56, 20.96) | 21.08 | 0.26 | (20.96, 21.21) | 20.76 | 0.42 | (20.56, 20.96) | 17.59 | 0.32 | (17.44, 17.74) | 20.6 | 0.49 | (20.36, 20.84) |
| TR | 60.11 | 0.83 | (59.71, 60.51) | 60.2 | 0.66 | (59.88, 60.51) | 58.14 | 0.86 | (57.72, 58.55) | 36.71 | 0.27 | (36.58, 36.84) | 59.84 | 0.56 | (59.57, 60.11) |
| MTF-MTF -EOMA | 18.58 | 2.31 | (17.47, 19.69) | 19.43 | 2.29 | (18.33, 20.53) | 20.34 | 2.5 | (19.14, 21.54) | 22.7 | 0.45 | (22.48, 22.92) | 18.87 | 2.77 | (17.54, 20.2) |
| MTF-TR -EOMA | 40.92 | 0.93 | (40.47, 41.36) | 41.25 | 0.72 | (40.91, 41.6) | 40.3 | 1.21 | (39.72, 40.88) | 32.53 | 1.17 | (31.97, 33.09) | 40.78 | 1.11 | (40.24, 41.31) |
| TR-MTF -EOMA | 19.52 | 1.31 | (18.53, 20.51) | 19.84 | 2.52 | (17.94, 21.75) | 19.99 | 2.34 | (18.23, 21.75) | 22.81 | 0.75 | (22.24, 23.37) | 19.86 | 2.27 | (18.15, 21.57) |
| TR-TR -EOMA | 41.07 | 0.84 | (40.67, 41.48) | 40.87 | 1.03 | (40.38, 41.37) | 40.79 | 1.12 | (40.26, 41.33) | 32.01 | 1.29 | (31.39, 32.63) | 40.91 | 1.2 | (40.33, 41.49) |
| MTF | 20.95 | 0.2 | (20.86, 21.05) | 21.04 | 0.14 | (20.97, 21.1) | 20.78 | 0.2 | (20.68, 20.88) | 17.65 | 0.15 | (17.58, 17.72) | 20.69 | 0.26 | (20.56, 20.81) |
| TR | 60.4 | 0.37 | (60.22, 60.58) | 60.61 | 0.33 | (60.45, 60.77) | 58.58 | 0.32 | (58.43, 58.73) | 37.92 | 0.26 | (37.79, 38.04) | 59.88 | 0.36 | (59.71, 60.06) |
| MTF-MTF -EOMA | 20.93 | 1.58 | (20.17, 21.69) | 21.11 | 1.14 | (20.57, 21.66) | 21.62 | 1.18 | (21.05, 22.18) | 22.95 | 0.29 | (22.81, 23.09) | 20.96 | 1.36 | (20.31, 21.61) |
| MTF-TR -EOMA | 40.4 | 0.41 | (40.2, 40.59) | 40.74 | 0.41 | (40.54, 40.93) | 39.89 | 0.59 | (39.6, 40.17) | 31.85 | 0.65 | (31.53, 32.16) | 40.33 | 0.43 | (40.12, 40.53) |
| TR-MTF -EOMA | 21.48 | 1.27 | (20.52, 22.45) | 21.72 | 1.37 | (20.68, 22.75) | 21.91 | 1.04 | (21.13, 22.69) | 23.0 | 0.29 | (22.78, 23.21) | 21.8 | 1.06 | (21.0, 22.6) |
| TR-TR -EOMA | 40.48 | 0.49 | (40.25, 40.72) | 40.57 | 0.34 | (40.4, 40.73) | 39.99 | 0.47 | (39.76, 40.22) | 31.36 | 0.62 | (31.06, 31.66) | 40.26 | 0.49 | (40.02, 40.49) |

Table 2: Experimental results displaying the average number of accesses for the hierarchical schemes that included the EOMA and the stand-alone schemes in MSEs, where $\alpha = 0.9$ and $k = 32$. For the hierarchical schemes, a list of size **128** was split into **32** sublists with **4** records each. The **top** portion of the table shows the asymptotic cost, while the amortized cost is at the **bottom**.

By way of example, consider Table 2, where the number of partitions $k = 32$. Observe that for the Zipf distribution, the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA yielded asymptotic costs of (18.58, 40.92, 19.52, 41.07) and amortized costs of (20.93, 40.40, 21.48, 40.48). As opposed to this, the stand-alone MTF and TR rule had asymptotic costs of (20.76, 60.11) and an amortized cost of (20.95, 60.40) respectively. These results corroborate the observation that the MTF-MTF-EOMA and the TR-MTF-EOMA schemes performed better than the MTF and TR in all the dependent query Environments under consideration. In contrast, the MTF scheme was still superior to the MTF-TR-EOMA and the TR-TR-EOMA.

To observe the performance of the hierarchical schemes in comparison to the MTF when the number of sub-lists increased, we refer the reader to Figures 5 - 8, which display the ratio of the asymptotic cost of the hierarchical schemes with the EOMA to the MTF scheme for different values of sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$. In the Figures, where asymptotic cost ratio is greater than unity, the graph indicates that the MTF results were superior to the corresponding hierarchical EOMA schemes. In contrast, results less than unity show that the hierarchical EOMA scheme is superior to the MTF rule, which was the case that occurred most of the time.

From Figure 5, we observed that the MTF-MTF-EOMA performed better than the MTF for sub-lists with $k \in \{2, 4, 8, 10, 16, 32\}$ for the Zipf, Linear and 80-20 distributions. However, when $k = 64$, the performance

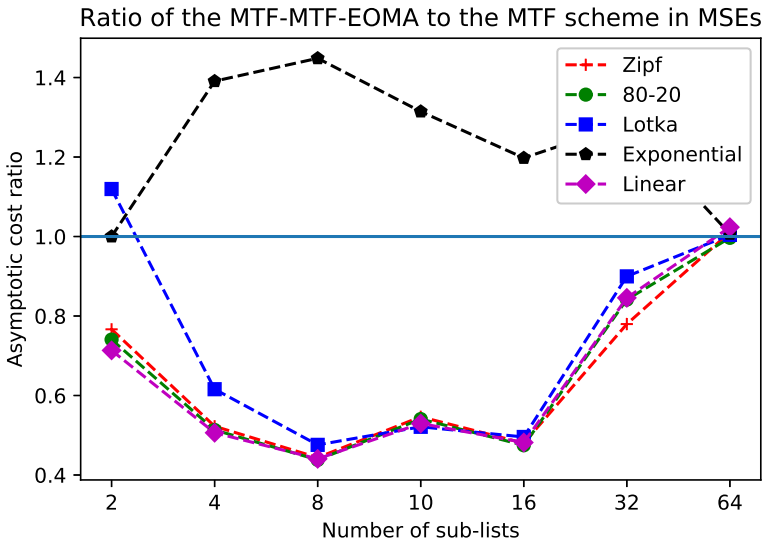



Figure 5: The asymptotic cost ratio of the MTF-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

of the MTF-MTF-EOMA across all distributions were at par with the MTF. The observation for the performance of the TR-MTF-EOMA against the MTF was similar to that of the MTF-MTF-EOMA (see Figure 6).

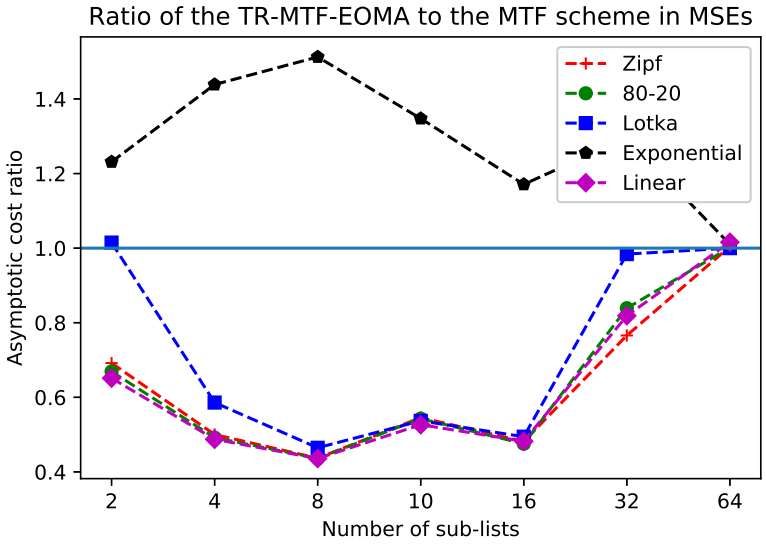



Figure 6: The asymptotic cost ratio of the TR-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

In Figure 7, we observed that the MTF-TR-EOMA scheme performed better than the MTF rule for the Zipf, Lotka, Exponential and Linear distributions when $k \in \{4, 8, 10\}$. For the Exponential distribution, the MTF rule was consistently superior to the MTF-TR-EOMA scheme for all sub-list sizes. When $k = 32$ and for all query distributions, the MTF was superior to the MTF-TR-EOMA. However, at $k = 64$, a strange phenomenon occurred, where the performance of the MT-TR-EOMA was comparable with the MTF. This is

after observing a superior MTF performance at $k = 32$. As a result, we observed a triangular shape between $k = 16$ and $k = 64$. The observation for the performance of the TR-TR-EOMA against the MTF was similar to that of the MTF-TR-EOMA in Figure 8 as they both had the TR as the outer-list reorganization rule.

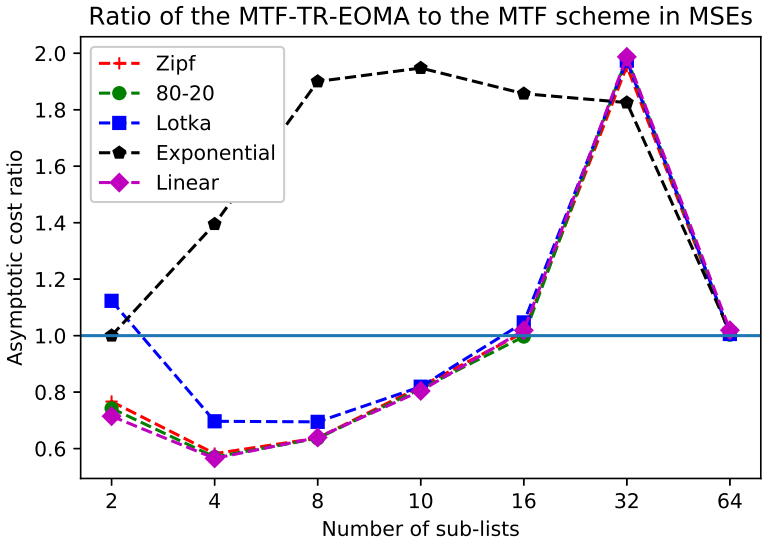



Figure 7: The asymptotic cost ratio of the MTF-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

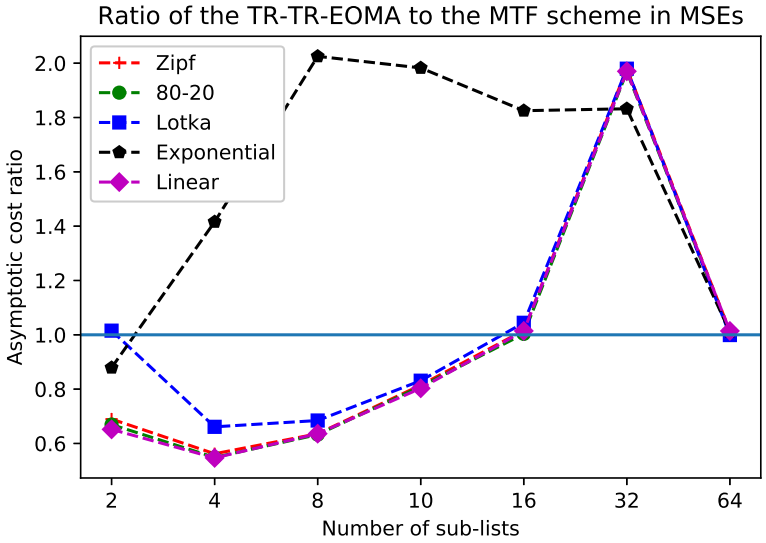



Figure 8: The asymptotic cost ratio of the TR-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

**Behavior with Environments' degree of dependence**: To discuss the behavior of the hierarchical schemes, which included the EOMA in the MSEs, we considered their performance with respect to the Environments' degree of dependence. Observe that the "Locality of Reference" constant, $\alpha$, controls the probabilistic measure that a query pair comes from the same query Environment or sub-list. Figure 9 shows the changes in the asymptotic cost of the stand-alone and hierarchical schemes that included the EOMA in MSEs as the

degree of dependence increased from a weak dependence, $\alpha = 0.1$, to a strong dependence, $\alpha = 0.9$.

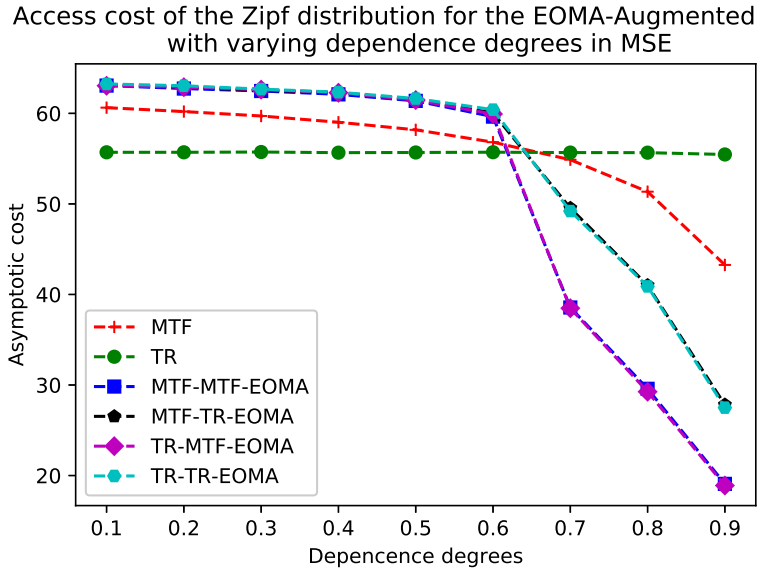



Figure 9: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with EOMA in the MSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

From the results in Figure 9, we observed that the performance of the hierarchical schemes with EOMA performed poorly against the MTF and TR schemes when the dependence degree was weak, i.e., effectively arriving from noisy or almost-random query accesses. However, when the dependence degree $\alpha$ was greater than $0.6$, we observed the reverse performance with the hierarchical EOMA-based schemes, which had demonstrably superior performances to the MTF and TR schemes. This empirical observation corroborates with the results reported by the authors of [5].

Figure 9 illustrates this phenomenon for the Zipf distribution. This observation was identical for the 80-20, Lotka, Exponential and Linear distributions, for a list of 128 elements partitioned into 8 sub-lists. The partition $k = 8$ was selected for a list of size 128, because, the schemes under consideration achieved, on average, their best performance at this juncture, as seen from the empirical results observed earlier. (Please see Figures 5 - 8).

Also observe, that as earlier noted, the MTF-MTF-EOMA and the TR-MTF-EOMA had comparable performances, and was superior to the MTF-TR-EOMA and the TR-TR-EOMA. In general, we confirm that the re-organization scheme is superior when it has the MTF rule as the outer list-context as opposed to the TR.

**Observations for the amortized costs**: To display the rate of the convergence of the schemes under consideration, Figure 10 shows the trend in the amortized cost of the first 100,000 queries for both the hierarchical schemes augmented with EOMA, and the stand-alone schemes. The perspective of how the re-organization schemes minimized the amortized cost is crucial for accessing their optimality in NSEs.

In Figure 10, it is easy to observe that from the first few queries, all the EOMA-augmented hierarchical schemes perform better than the TR rule in minimizing the amortized cost. Right about the $10,000^{th}$ query, the EOMA-augmented hierarchical schemes catches-up with the MTF rule in terms of performance and from thereon boasts a far superior performance compared to the MTF. A key observation is that the EOMA-

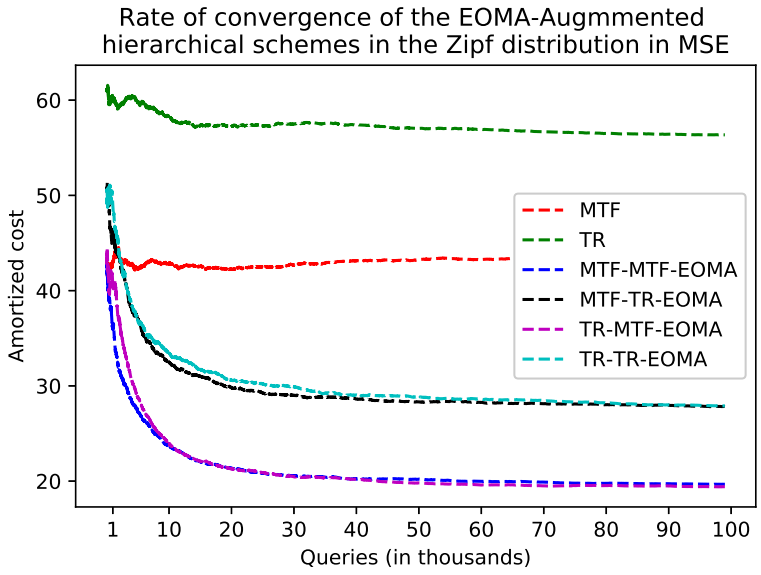



Figure 10: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes augmented with the EOMA in a MSE with $\alpha = 0.9$ and $k = 8$.

augmented hierarchical schemes appeared to converge after about 30,000 queries. As opposed to this, the MTF and TR schemes quickly plateaued with no additional gains in performance with extended interactions with the Environment. Although Figure 10 shows the rate of convergence for the Zipf distribution, the observed phenomena were similar for the 80-20, Lotka, Exponential and Linear distributions.

Tables 3 and 4 compares the results of the EOMA-augmented hierarchical schemes with the OMA-augmented hierarchical schemes and stand-alone schemes when $k = 2$ and $k = 32$ respectively by reporting their asymptotic costs in MSEs. From Table 3, when the sublist size is larger (i.e. $k = 2$), the EOMA-augmented schemes are generally superior to the OMA-augmented variant. The superiority shows more when we consider the L-shaped Exponential and Lotka distributions.

As an example, consider the Exponential distribution where the OMA-hierarchical schemes have the asymptotic costs (11.95, 10.37, 10.67, 13.16) for the MTF-MTF-OMA, MTF-TR-OMA, TR-MTF-OMA and TR-TR-OMA respectively. As opposed to this, the EOMA-hierarchical schemes have the asymptotic costs (3.3, 3.8, 2.48, 2.96) for the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA schemes respectively, which are markedly superior.

However, it turns out that when the sublists are smaller as shown in Table 4, where $k = 32$, the results of the EOMA-augmented schemes and the OMA-augmented schemes are comparable with varying degrees of performances. Nonetheless, we observed that the EOMA-augmented schemes are still generally superior in schemes that have the TR rule as the outer-list context. This mitigates, the earlier observed degradation in performance in schemes when the outer-list context is the TR rule when the number of partitions increased in Environments exhibiting "locality of reference".

For example, in the Linear distribution, the asymptotic cost for the MTF-TR-OMA and TR-TR-OMA schemes were (42.98, 45.66) with standard deviation and confidence intervals (SD: 1.09, CI: [42.46, 43.51])

and (SD:1.53, CI: [44.93, 46.40]) respectively, while that of the MTF-TR-EOMA and TR-TR-EOMA hierarchical schemes were (40.78, 40.91) with standard deviation and confidence intervals of (SD: 1.11, CI: [40.24, 41.31]) and (SD:1.20, CI: [40.33, 41.49]) respectively. These results, as compared to the TR rule in the same distribution has an asymptotic cost of 59.84 with SD (0.56) and CI ([59.57, 60.11]). Clearly, we can see the superiority of the EOMA-hierarchical schemes in this context.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 45.85 | 0.28 | (45.71, 45.98) | 49.66 | 0.29 | (49.52, 49.8) | 24.17 | 0.31 | (24.02, 24.32) | 2.81 | 0.03 | (2.79, 2.83) | 54.84 | 0.29 | (54.7, 54.98) |
| TR | 37.54 | 0.23 | (37.44, 37.65) | 41.39 | 0.29 | (41.25, 41.53) | 19.11 | 0.26 | (18.99, 19.24) | 2.54 | 0.04 | (2.52, 2.56) | 48.29 | 0.28 | (48.16, 48.43) |
| MTF-MTF -OMA | 36.1 | 0.5 | (35.85, 36.34) | 37.33 | 0.29 | (37.19, 37.48) | 29.84 | 0.77 | (29.47, 30.21) | 11.95 | 3.02 | (10.5, 13.4) | 40.21 | 0.44 | (40.0, 40.42) |
| MTF-TR -OMA | 36.16 | 0.5 | (35.92, 36.41) | 37.52 | 0.42 | (37.32, 37.72) | 29.48 | 0.74 | (29.12, 29.83) | 10.37 | 3.31 | (8.78, 11.95) | 40.22 | 0.45 | (40.0, 40.43) |
| TR-MTF -OMA | 32.7 | 0.37 | (32.52, 32.88) | 34.32 | 0.42 | (34.12, 34.52) | 27.49 | 0.95 | (27.03, 27.94) | 10.67 | 3.17 | (9.15, 12.19) | 36.89 | 0.6 | (36.61, 37.18) |
| TR-TR -OMA | 33.66 | 2.91 | (32.27, 35.06) | 36.51 | 5.29 | (33.98, 39.05) | 27.56 | 0.8 | (27.17, 27.94) | 13.16 | 2.45 | (11.99, 14.34) | 36.99 | 0.34 | (36.83, 37.15) |
| MTF-MTF -EOMA | 35.13 | 0.25 | (35.01, 35.26) | 36.84 | 0.24 | (36.72, 36.95) | 27.2 | 0.43 | (27.0, 27.41) | 3.3 | 2.1 | (2.29, 4.31) | 39.25 | 0.26 | (39.12, 39.38) |
| MTF-TR -EOMA | 35.19 | 0.28 | (35.05, 35.32) | 36.91 | 0.37 | (36.74, 37.09) | 27.1 | 0.33 | (26.94, 27.26) | 3.8 | 2.96 | (2.38, 5.22) | 39.27 | 0.23 | (39.17, 39.38) |
| TR-MTF -EOMA | 31.6 | 0.34 | (31.34, 31.85) | 33.36 | 0.29 | (33.14, 33.59) | 24.45 | 0.25 | (24.26, 24.64) | 2.48 | 0.03 | (2.45, 2.5) | 35.73 | 0.19 | (35.58, 35.87) |
| TR-TR -EOMA | 31.6 | 0.29 | (31.46, 31.74) | 33.14 | 0.29 | (33.0, 33.28) | 24.69 | 0.64 | (24.39, 25.0) | 2.96 | 2.01 | (1.99, 3.92) | 35.77 | 0.28 | (35.64, 35.9) |

Table 3: Experimental results comparing the asymptotic costs for the OMA and EOMA-augmented hierarchical schemes with the stand-alone schemes in MSEs, where $\alpha = 0.9$ and $k = 2$. The **top** portion of the table shows the stand-alone schemes, while the **middle** and **bottom** portions the OMA and EOMA-augmented hierarchical schemes respectively.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 20.76 | 0.41 | (20.56, 20.96) | 21.08 | 0.26 | (20.96, 21.21) | 20.76 | 0.42 | (20.56, 20.96) | 17.59 | 0.32 | (17.44, 17.74) | 20.6 | 0.49 | (20.36, 20.84) |
| TR | 60.11 | 0.83 | (59.71, 60.51) | 60.2 | 0.66 | (59.88, 60.51) | 58.14 | 0.86 | (57.72, 58.55) | 36.71 | 0.27 | (36.58, 36.84) | 59.84 | 0.56 | (59.57, 60.11) |
| MTF-MTF -OMA | 14.41 | 0.76 | (14.04, 14.77) | 14.56 | 0.73 | (14.21, 14.92) | 14.79 | 0.61 | (14.49, 15.08) | 19.54 | 0.78 | (19.16, 19.91) | 14.84 | 0.66 | (14.52, 15.15) |
| MTF-TR -OMA | 42.36 | 1.59 | (41.6, 43.12) | 42.63 | 1.3 | (42.01, 43.26) | 42.26 | 1.06 | (41.75, 42.77) | 41.13 | 0.87 | (40.71, 41.55) | 42.98 | 1.09 | (42.46, 43.51) |
| TR-MTF -OMA | 14.31 | 0.65 | (14.0, 14.62) | 14.71 | 0.81 | (14.32, 15.1) | 14.66 | 0.7 | (14.33, 15.0) | 20.19 | 0.85 | (19.78, 20.6) | 15.13 | 0.7 | (14.79, 15.46) |
| TR-TR -OMA | 44.47 | 1.69 | (43.66, 45.29) | 45.04 | 1.81 | (44.17, 45.91) | 45.39 | 1.33 | (44.75, 46.02) | 40.92 | 1.63 | (40.14, 41.7) | 45.66 | 1.53 | (44.93, 46.4) |
| MTF-MTF -EOMA | 18.58 | 2.31 | (17.47, 19.69) | 19.43 | 2.29 | (18.33, 20.53) | 20.34 | 2.5 | (19.14, 21.54) | 22.7 | 0.45 | (22.48, 22.92) | 18.87 | 2.77 | (17.54, 20.2) |
| MTF-TR -EOMA | 40.92 | 0.93 | (40.47, 41.36) | 41.25 | 0.72 | (40.91, 41.6) | 40.3 | 1.21 | (39.72, 40.88) | 32.53 | 1.17 | (31.97, 33.09) | 40.78 | 1.11 | (40.24, 41.31) |
| TR-MTF -EOMA | 19.52 | 1.31 | (18.53, 20.51) | 19.84 | 2.52 | (17.94, 21.75) | 19.99 | 2.34 | (18.23, 21.75) | 22.81 | 0.75 | (22.24, 23.37) | 19.86 | 2.27 | (18.15, 21.57) |
| TR-TR -EOMA | 41.07 | 0.84 | (40.67, 41.48) | 40.87 | 1.03 | (40.38, 41.37) | 40.79 | 1.12 | (40.26, 41.33) | 32.01 | 1.29 | (31.39, 32.63) | 40.91 | 1.2 | (40.33, 41.49) |

Table 4: Experimental results comparing the asymptotic costs for the OMA and EOMA-augmented hierarchical schemes with the stand-alone schemes in MSEs, where $\alpha = 0.9$ and $k = 32$. The **top** portion of the table shows the stand-alone schemes, while the **middle** and **bottom** portions the OMA and EOMA-augmented hierarchical schemes respectively.

# 9 Results and Discussions: Performance in PSEs

This section reports the performance of the EOMA-augmented hierarchical schemes in the PSE. The experimental setup was the same as in the case of the MSE reported above, but with the period $T = 30$. The variable $T$ is a hyper-parameter, and its value was chosen empirically to allow for a sufficient number of queries to arrive from the query space before switching to another pattern.

**Behavior for Exponential Distributions**: Table 5 compares the performance of the EOMA-augmented hierarchical schemes with the stand-alone MTF and TR schemes in PSEs when the number of sub-lists $k = 2$. From Table 5, we see that the hierarchical schemes with the EOMA performed better than their stand-alone counterparts, except for the Exponential distribution where the MTF and TF rules are almost at par or even better than the EOMA-augmented schemes. As an example, to confirm this observation, consider the 80-20 distribution. Here, the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA had asymptotic and amortized costs of (29.97, 29.99, 26.45, 26.43) and (30.23, 30.14, 27.03, 26.97) respectively, while the MTF and TR schemes had an asymptotic and amortized cost of (50.38, 41.46) and (50.53, 42.62). Clearly, the EOMA-augmented hierarchical schemes were superior to the stand-alone schemes. The standard deviation and confidence interval estimates are shown in the corresponding table.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 46.52 | 0.25 | (46.4, 46.64) | 50.38 | 0.37 | (50.2, 50.56) | 24.22 | 0.18 | (24.13, 24.31) | 2.52 | 0.03 | (2.51, 2.53) | 56.23 | 0.17 | (56.15, 56.31) |
| TR | 37.43 | 0.21 | (37.33, 37.53) | 41.46 | 0.35 | (41.29, 41.63) | 19.02 | 0.12 | (18.96, 19.08) | 2.51 | 0.04 | (2.49, 2.52) | 48.31 | 0.23 | (48.2, 48.42) |
| MTF-MTF -EOMA | 28.38 | 0.14 | (28.31, 28.44) | 29.97 | 0.12 | (29.91, 30.03) | 20.4 | 0.07 | (20.36, 20.43) | 2.52 | 0.03 | (2.5, 2.53) | 32.46 | 0.15 | (32.39, 32.53) |
| MTF-TR -EOMA | 28.34 | 0.12 | (28.28, 28.4) | 29.99 | 0.15 | (29.92, 30.07) | 20.3 | 0.11 | (20.25, 20.35) | 2.68 | 0.72 | (2.33, 3.02) | 32.46 | 0.14 | (32.4, 32.53) |
| TR-MTF -EOMA | 24.84 | 0.15 | (24.73, 24.95) | 26.45 | 0.14 | (26.35, 26.55) | 17.64 | 0.09 | (17.57, 17.71) | 2.46 | 0.03 | (2.43, 2.48) | 29.03 | 0.13 | (28.93, 29.13) |
| TR-TR -EOMA | 24.83 | 0.15 | (24.76, 24.9) | 26.43 | 0.14 | (26.36, 26.5) | 17.65 | 0.07 | (17.61, 17.68) | 2.43 | 0.02 | (2.42, 2.44) | 28.99 | 0.14 | (28.93, 29.06) |
| MTF | 46.55 | 0.12 | (46.49, 46.61) | 50.53 | 0.13 | (50.46, 50.59) | 24.23 | 0.07 | (24.2, 24.26) | 2.54 | 0.01 | (2.53, 2.55) | 56.26 | 0.1 | (56.21, 56.3) |
| TR | 38.74 | 0.21 | (38.64, 38.84) | 42.62 | 0.18 | (42.53, 42.7) | 20.31 | 0.12 | (20.25, 20.37) | 2.95 | 0.09 | (2.9, 2.99) | 49.53 | 0.19 | (49.44, 49.62) |
| MTF-MTF -EOMA | 28.63 | 0.2 | (28.54, 28.73) | 30.23 | 0.19 | (30.13, 30.32) | 21.12 | 0.43 | (20.91, 21.32) | 2.54 | 0.01 | (2.54, 2.55) | 32.6 | 0.09 | (32.56, 32.64) |
| MTF-TR -EOMA | 28.7 | 0.22 | (28.59, 28.8) | 30.14 | 0.12 | (30.09, 30.2) | 21.2 | 0.4 | (21.01, 21.39) | 2.7 | 0.71 | (2.36, 3.05) | 32.61 | 0.09 | (32.57, 32.66) |
| TR-MTF -EOMA | 25.4 | 0.36 | (25.13, 25.67) | 27.03 | 0.27 | (26.82, 27.23) | 19.75 | 1.47 | (18.64, 20.87) | 2.64 | 0.01 | (2.63, 2.65) | 29.35 | 0.13 | (29.25, 29.44) |
| TR-TR -EOMA | 25.66 | 0.43 | (25.46, 25.87) | 26.97 | 0.23 | (26.86, 27.08) | 19.55 | 0.61 | (19.25, 19.84) | 2.64 | 0.02 | (2.63, 2.65) | 29.36 | 0.11 | (29.31, 29.42) |

Table 5: Experimental results between the hierarchical schemes with EOMA and the stand-alone schemes in a Periodic Switching Environment with period $T = 30$ and $k = 2$. The **top** portion of the table shows the asymptotic cost, while the amortized cost is at the **bottom**. For the hierarchical schemes, a list of size **128** is split into **2** sublists with **64** records each.

For the L-shaped Exponential distribution, where the "de-facto" stand-alone schemes generally have superior performances, the MTF and TR schemes have asymptotic and amortized cost of (2.52, 2.54) and (2.51, 2.95), while the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA had corresponding costs of (2.52, 2.68, 2.46, 2.43) and (2.54, 2.70, 2.64, 2.64) respectively. From this example, we see that the

MTF and TR stand-alone schemes have comparable performances to the EOMA-augmented hierarchical schemes. This conclusion is further supported by the standard deviation and confidence interval estimates shown in the corresponding table.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 18.02 | 0.02 | (18.01, 18.03) | 18.01 | 0.02 | (18.0, 18.02) | 17.91 | 0.02 | (17.9, 17.92) | 16.47 | 0.11 | (16.42, 16.53) | 17.75 | 0.01 | (17.74, 17.76) |
| TR | 61.9 | 0.18 | (61.81, 61.98) | 62.32 | 0.31 | (62.17, 62.47) | 60.02 | 0.26 | (59.89, 60.14) | 39.07 | 0.26 | (38.95, 39.2) | 61.47 | 0.25 | (61.35, 61.6) |
| MTF-MTF -EOMA | 10.23 | 0.48 | (10.0, 10.46) | 10.13 | 0.46 | (9.91, 10.35) | 10.39 | 0.61 | (10.1, 10.69) | 16.3 | 1.21 | (15.72, 16.88) | 10.37 | 0.61 | (10.08, 10.66) |
| MTF-TR -EOMA | 65.54 | 1.31 | (64.91, 66.17) | 65.97 | 0.65 | (65.66, 66.29) | 63.92 | 3.03 | (62.46, 65.37) | 21.19 | 1.49 | (20.47, 21.9) | 65.27 | 1.3 | (64.65, 65.89) |
| TR-MTF -EOMA | 13.49 | 0.48 | (13.13, 13.85) | 13.51 | 0.89 | (12.84, 14.18) | 13.61 | 0.48 | (13.25, 13.97) | 18.21 | 0.72 | (17.67, 18.76) | 13.35 | 0.41 | (13.04, 13.66) |
| TR-TR -EOMA | 65.63 | 0.87 | (65.21, 66.04) | 65.22 | 1.92 | (64.3, 66.14) | 64.33 | 1.72 | (63.51, 65.16) | 21.12 | 1.42 | (20.44, 21.81) | 64.75 | 2.02 | (63.78, 65.72) |
| MTF | 17.97 | 0.01 | (17.97, 17.97) | 17.98 | 0.01 | (17.97, 17.98) | 17.87 | 0.02 | (17.86, 17.87) | 16.44 | 0.05 | (16.42, 16.46) | 17.71 | 0.01 | (17.71, 17.71) |
| TR | 62.35 | 0.17 | (62.26, 62.43) | 62.55 | 0.12 | (62.49, 62.6) | 60.51 | 0.17 | (60.43, 60.59) | 40.24 | 0.14 | (40.17, 40.31) | 61.86 | 0.19 | (61.76, 61.95) |
| MTF-MTF -EOMA | 13.33 | 0.53 | (13.08, 13.59) | 13.29 | 0.62 | (12.99, 13.59) | 13.59 | 0.55 | (13.33, 13.85) | 18.42 | 0.86 | (18.01, 18.84) | 13.35 | 0.62 | (13.05, 13.65) |
| MTF-TR -EOMA | 55.91 | 1.86 | (55.01, 56.8) | 54.84 | 1.43 | (54.15, 55.53) | 53.17 | 3.47 | (51.5, 54.83) | 20.87 | 0.66 | (20.56, 21.19) | 55.17 | 2.17 | (54.13, 56.21) |
| TR-MTF -EOMA | 13.49 | 0.48 | (13.13, 13.85) | 13.51 | 0.89 | (12.84, 14.18) | 13.61 | 0.48 | (13.25, 13.97) | 18.21 | 0.72 | (17.67, 18.76) | 13.35 | 0.41 | (13.04, 13.66) |
| TR-TR -EOMA | 55.2 | 2.26 | (54.11, 56.28) | 55.73 | 2.05 | (54.75, 56.72) | 52.78 | 2.4 | (51.63, 53.93) | 20.73 | 0.46 | (20.51, 20.95) | 55.07 | 1.81 | (54.2, 55.94) |

Table 6: Experimental results between the hierarchical schemes with EOMA and the stand-alone schemes in a Periodic Switching Environment with period $T = 30$ and $k = 32$. The **top** portion of the table shows the asymptotic cost, while the amortized cost is at the **bottom**. For the hierarchical schemes, a list of size **128** is split into **32** sublists with **4** records each.

**Behavior when increasing the number of partitions**: We now report the results for the scenario when the number of partitions increased, and $k = 32$. Table 6 highlights the performance of the EOMA-augmented hierarchical schemes and the stand-alone schemes. From this table, we can make an interesting observation, i.e., that while the MTF-MTF-EOMA and the TR-MTF-EOMA performed better than their hierarchical counterparts *and* the MTF and TR schemes, the MTF-TR-EOMA and the TR-TR-EOMA returned poor results in comparison to all the other schemes. This was true in all dependent models except for the Exponential distribution where the TR rule was mostly inferior. This resonates with what we noted earlier about how schemes whose re-organization strategy for the outer list context was TR possessed a poorer performance than those whose outer-context was the MTF rule. This phenomenon was exacerbated in PSEs.

As an example, consider the Exponential distribution where the asymptotic and amortized costs for the MTF and TR were (16.47, 39.07) and (16.44, 40.24) respectively. The asymptotic and amortized costs for the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA were (16.30, 21.19, 18.21, 21.12) and (18.42, 20.87, 18.21, 20.73) respectively. The reader should particularly note the inferior performance of the EOMA-augmented hierarchical schemes in which the TR was the outer-list context, as opposed to those where the MTF was the outer list context.

We now report another perspective on the performance of the hierarchical schemes in PSEs as the number of sub-lists, $k$, increases. Figures 11 - 14 show the ratio of the asymptotic cost of the hierarchical schemes

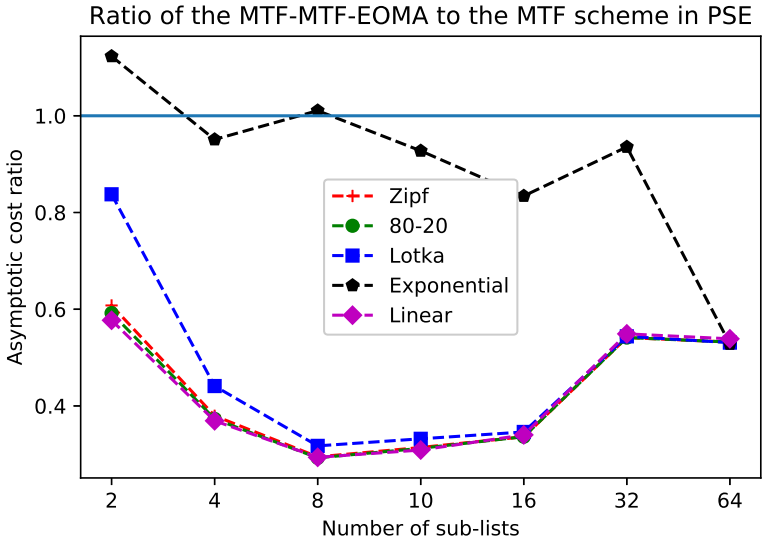



Figure 11: The asymptotic cost ratio of the MTF-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

augmented with the EOMA to the MTF scheme for varying sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$, and where the period $T = 30$. In the named figures, a ratio greater than unity indicates that the MTF rule was superior. In the contrary case, if this index is less than unity, the hierarchical scheme is superior.

From Figure 11, we observe that the MTF-MTF-EOMA performed better than the MTF for all sub-list variations in the Zipf, 80-20, Lotka, and Linear distributions, except for the Exponential, where it was comparable to the MTF. It became significantly superior when $k > 8$. The observation for the performance of the MTF-MTF-EOMA against the MTF was similar to that of the TR-MTF-EOMA, as seen in Figure 12.

From Figure 13, we see that the MTF-TR-EOMA scheme performed better than the MTF rule for the Zipf, 80-20, Lotka, and Linear distributions when $k = \{2, 4, 8, 10\}$. However, as in the MSE case, we observed the same triangular shape between $k = 16$ and $k = 64$. The Zipf, 80-20, Lotka, and Linear distributions performed poorly in comparison with the MTF at $k = 32$ and then became superior when $k = 64$. The performance of the MTF-TR-EOMA against the MTF was similar to the TR-TR-EOMA displayed in Figure 14.

Figure 15 shows that for periods $T$ greater than 10, the hierarchical schemes augmented with the EOMA have better asymptotic costs compared to the stand-alone MTF and TR schemes. Also, for small numbers of queries arriving from a query-space $Q_i$ before switching to another query-space $Q_j$ (e.g., for $T = 10$), although the MTF and TR rules performed marginally better than the MTF-TR-EOMA and TR-TR-EOMA, the MTF-MTF-EOMA and TR-MTF-EOMA still boasted superior performances.

From Figure 16 we observe that the hierarchical schemes with the EOMA began to converge very early in the query stream and that they did better than the MTF and TR schemes at the initial stages of receiving queries from the Environment. Also, while the stand-alone schemes appeared to hit a flat-line and achieve no further improvement in performance as the number of queries increased, the hierarchical schemes augmented with the EOMA, on the other hand, decreased rapidly and converged after about the $10,000^{th}$ query.

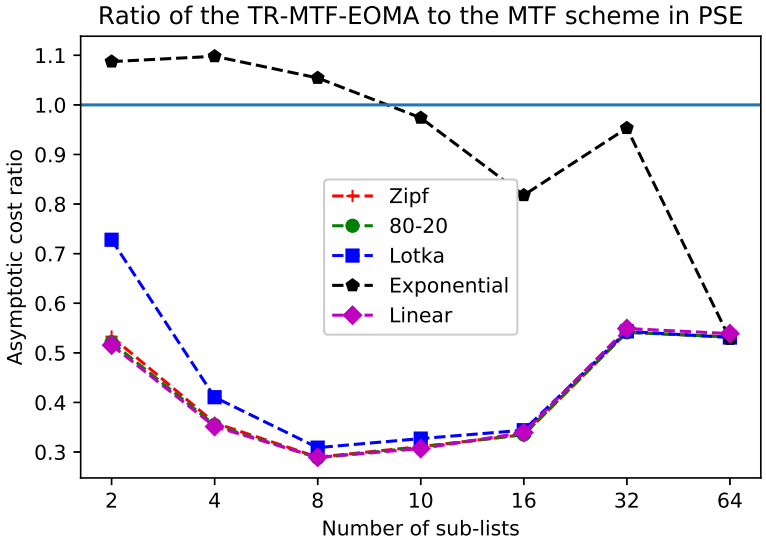



Figure 12: The asymptotic cost ratio of the TR-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

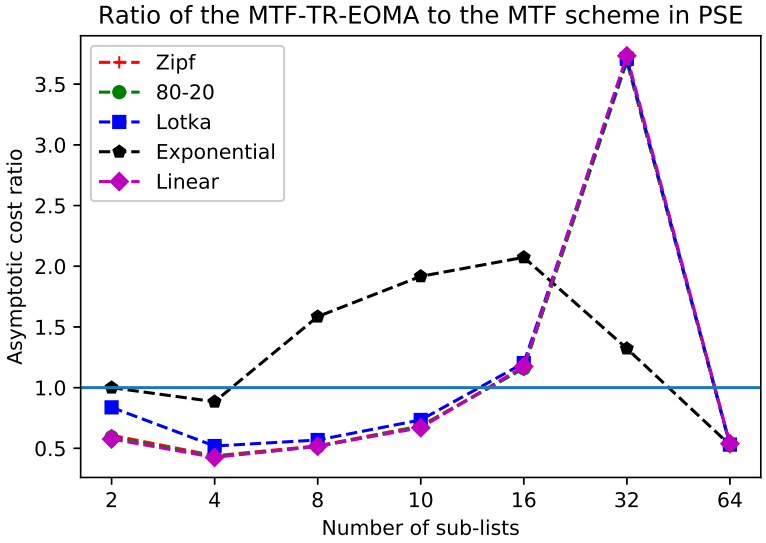



Figure 13: The asymptotic cost ratio of the MTF-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

Tables 7 and 8 compare the results of the EOMA-augmented hierarchical schemes with the OMA-augmented hierarchical schemes and stand-alone schemes when $k = 2$ and $k = 32$ respectively by reporting their asymptotic costs in PSEs. From Table 7, similar to the MSE case, when the sublist size is larger (i.e. $k = 2$), the EOMA-augmented schemes are generally superior to the OMA-augmented variant. Also, as in the MSE, the superiority is highlighted when we consider the L-shaped Exponential and Lotka distributions.

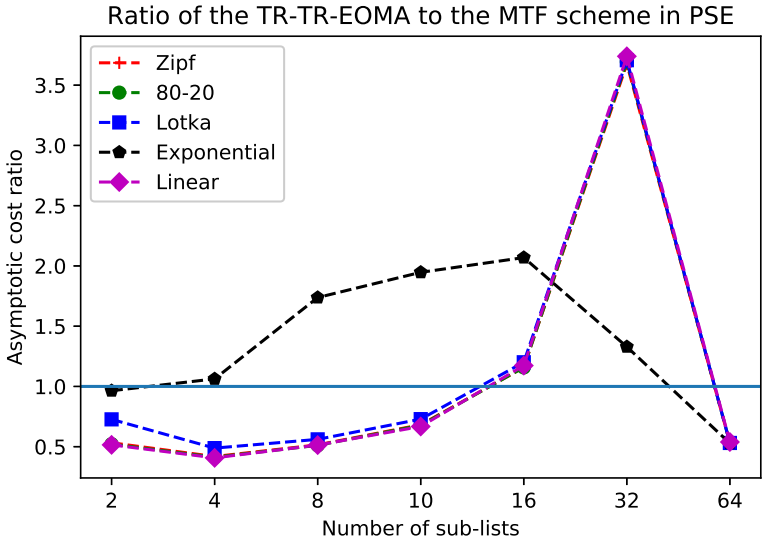



Figure 14: The asymptotic cost ratio of the TR-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

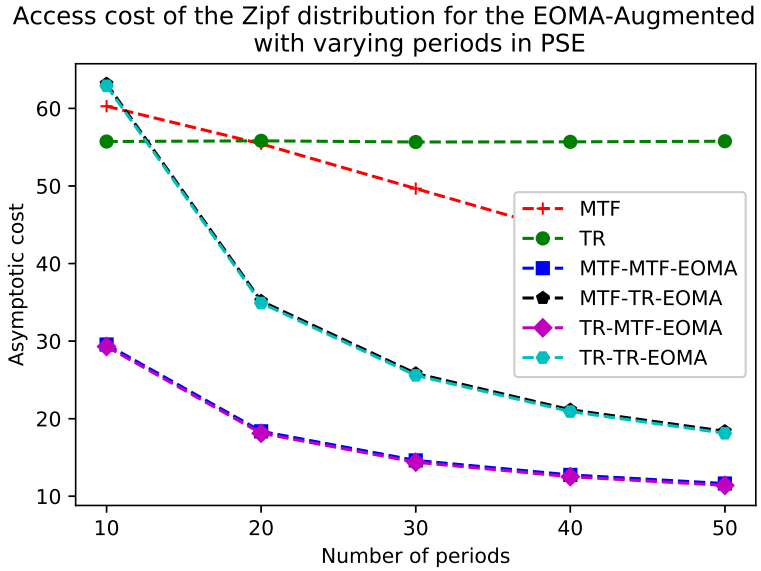



Figure 15: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with EOMA in the PSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

As an example, consider the Exponential distribution where the OMA-hierarchical schemes have the asymptotic costs (6.91, 7.43, 8.10, 8.10) for the MTF-MTF-OMA, MTF-TR-OMA, TR-MTF-OMA and TR-TR-OMA respectively. Whereas, the EOMA-hierarchical schemes have the asymptotic costs (2.52, 2.68, 2.46, 2.43) for the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA schemes respectively. The superiority of the EOMA schemes is obvious!

However, it turns out that when the sublists are smaller as shown in Table 8, where $k = 32$, the re-

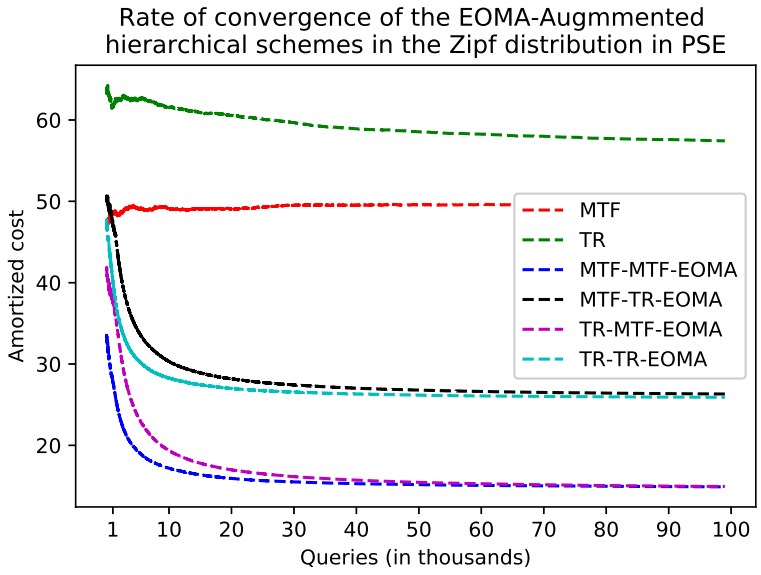



Figure 16: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with EOMA in the PSE with period $T = 30$ and $k = 8$.

sults of the EOMA-augmented schemes and the OMA-augmented schemes generally have comparable performances. As earlier mentioned, the case where the performances degrades in schemes that have the outer-list context as the TR rule is worsened in the PSE for both the OMA and EOMA-augmented hierarchical schemes. It is only in the Exponential distribution, where we observe a significant superiority of the EOMA-augmented schemes to their OMA-augmented counterparts.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 46.52 | 0.25 | (46.4, 46.64) | 50.38 | 0.37 | (50.2, 50.56) | 24.22 | 0.18 | (24.13, 24.31) | 2.52 | 0.03 | (2.51, 2.53) | 56.23 | 0.17 | (56.15, 56.31) |
| TR | 37.43 | 0.21 | (37.33, 37.53) | 41.46 | 0.35 | (41.29, 41.63) | 19.02 | 0.12 | (18.96, 19.08) | 2.51 | 0.04 | (2.49, 2.52) | 48.31 | 0.23 | (48.2, 48.42) |
| MTF-MTF -OMA | 30.79 | 0.85 | (30.39, 31.2) | 32.43 | 0.62 | (32.14, 32.73) | 26.14 | 1.31 | (25.52, 26.77) | 6.91 | 1.52 | (6.18, 7.64) | 34.58 | 0.63 | (34.28, 34.88) |
| MTF-TR -OMA | 31.21 | 0.91 | (30.78, 31.65) | 32.5 | 0.78 | (32.12, 32.87) | 26.34 | 1.11 | (25.81, 26.87) | 7.43 | 1.77 | (6.58, 8.28) | 34.72 | 0.66 | (34.4, 35.04) |
| TR-MTF -OMA | 28.18 | 0.93 | (27.73, 28.62) | 29.0 | 0.69 | (28.67, 29.33) | 24.19 | 1.28 | (23.58, 24.8) | 8.1 | 2.65 | (6.82, 9.37) | 31.53 | 0.56 | (31.26, 31.8) |
| TR-TR -OMA | 28.27 | 0.96 | (27.81, 28.74) | 29.26 | 0.77 | (28.89, 29.63) | 25.51 | 1.54 | (24.77, 26.25) | 8.1 | 3.06 | (6.63, 9.57) | 31.56 | 0.59 | (31.28, 31.85) |
| MTF-MTF -EOMA | 28.38 | 0.14 | (28.31, 28.44) | 29.97 | 0.12 | (29.91, 30.03) | 20.4 | 0.07 | (20.36, 20.43) | 2.52 | 0.03 | (2.5, 2.53) | 32.46 | 0.15 | (32.39, 32.53) |
| MTF-TR -EOMA | 28.34 | 0.12 | (28.28, 28.4) | 29.99 | 0.15 | (29.92, 30.07) | 20.3 | 0.11 | (20.25, 20.35) | 2.68 | 0.72 | (2.33, 3.02) | 32.46 | 0.14 | (32.4, 32.53) |
| TR-MTF -EOMA | 24.84 | 0.15 | (24.73, 24.95) | 26.45 | 0.14 | (26.35, 26.55) | 17.64 | 0.09 | (17.57, 17.71) | 2.46 | 0.03 | (2.43, 2.48) | 29.03 | 0.13 | (28.93, 29.13) |
| TR-TR -EOMA | 24.83 | 0.15 | (24.76, 24.9) | 26.43 | 0.14 | (26.36, 26.5) | 17.65 | 0.07 | (17.61, 17.68) | 2.43 | 0.02 | (2.42, 2.44) | 28.99 | 0.14 | (28.93, 29.06) |

Table 7: Experimental results comparing the asymptotic costs for the OMA and EOMA-augmented hierarchical schemes with the stand-alone schemes in PSEs, where $T = 30$ and $k = 2$. The **top** portion of the table shows the stand-alone schemes, while the **middle** and **bottom** portions the OMA and EOMA-augmented hierarchical schemes respectively.

| Scheme | Zipf | SD | CI - 95% | 80-20 | SD | CI - 95% | Lotka | SD | CI - 95% | Exp. | SD | CI - 95% | Linear | SD | CI - 95% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MTF | 18.02 | 0.02 | (18.01, 18.03) | 18.01 | 0.02 | (18.0, 18.02) | 17.91 | 0.02 | (17.9, 17.92) | 16.47 | 0.11 | (16.42, 16.53) | 17.75 | 0.01 | (17.74, 17.76) |
| TR | 61.9 | 0.18 | (61.81, 61.98) | 62.32 | 0.31 | (62.17, 62.47) | 60.02 | 0.26 | (59.89, 60.14) | 39.07 | 0.26 | (38.95, 39.2) | 61.47 | 0.25 | (61.35, 61.6) |
| MTF-MTF -OMA | 11.0 | 0.63 | (10.7, 11.3) | 10.87 | 0.58 | (10.6, 11.15) | 11.33 | 0.51 | (11.08, 11.57) | 15.33 | 0.84 | (14.92, 15.73) | 11.59 | 0.5 | (11.35, 11.83) |
| MTF-TR -OMA | 65.25 | 0.96 | (64.79, 65.71) | 65.11 | 1.38 | (64.45, 65.77) | 64.16 | 1.46 | (63.46, 64.86) | 48.89 | 4.31 | (46.82, 50.96) | 64.01 | 1.57 | (63.26, 64.77) |
| TR-MTF -OMA | 10.97 | 0.71 | (10.63, 11.31) | 11.09 | 0.56 | (10.82, 11.36) | 11.06 | 0.49 | (10.82, 11.3) | 15.55 | 0.69 | (15.22, 15.89) | 11.73 | 0.64 | (11.43, 12.04) |
| TR-TR -OMA | 60.55 | 0.98 | (60.08, 61.02) | 60.98 | 1.42 | (60.3, 61.66) | 59.77 | 1.51 | (59.04, 60.49) | 48.37 | 2.03 | (47.39, 49.34) | 60.45 | 1.55 | (59.7, 61.19) |
| MTF-MTF -EOMA | 10.23 | 0.48 | (10.0, 10.46) | 10.13 | 0.46 | (9.91, 10.35) | 10.39 | 0.61 | (10.1, 10.69) | 16.3 | 1.21 | (15.72, 16.88) | 10.37 | 0.61 | (10.08, 10.66) |
| MTF-TR -EOMA | 65.54 | 1.31 | (64.91, 66.17) | 65.97 | 0.65 | (65.66, 66.29) | 63.92 | 3.03 | (62.46, 65.37) | 21.19 | 1.49 | (20.47, 21.9) | 65.27 | 1.3 | (64.65, 65.89) |
| TR-MTF -EOMA | 13.49 | 0.48 | (13.13, 13.85) | 13.51 | 0.89 | (12.84, 14.18) | 13.61 | 0.48 | (13.25, 13.97) | 18.21 | 0.72 | (17.67, 18.76) | 13.35 | 0.41 | (13.04, 13.66) |
| TR-TR -EOMA | 65.63 | 0.87 | (65.21, 66.04) | 65.22 | 1.92 | (64.3, 66.14) | 64.33 | 1.72 | (63.51, 65.16) | 21.12 | 1.42 | (20.44, 21.81) | 64.75 | 2.02 | (63.78, 65.72) |

Table 8: Experimental results comparing the asymptotic costs for the OMA and EOMA-augmented hierarchical schemes with the stand-alone schemes in PSEs, where $T = 30$ and $k = 32$. The **top** portion of the table shows the stand-alone schemes, while the **middle** and **bottom** portions the OMA and EOMA-augmented hierarchical schemes respectively.

# 10 Results and Discussions: Periodic Variations

In Periodic Environments, the hierarchical schemes that incorporated the EOMA were able to boost their performance if they possessed an insight into the period, $T$, of the Environment. This implied that the schemes could preempt the EOMA's ordering by moving the first sublist to the end of the list after $T$ queries. This move was predicated on the observation that the elements from the completed query space would not be requested again until after $(k-1)T$ queries. Schemes with such a prior awareness of period $T$ are referred to by including the prefix "Periodic", leading to the MTF-MTF-EOMA-Periodic, MTF-TR-EOMA-Periodic, TR-MTF-EOMA-Periodic and TR-TR-EOMA-Periodic respectively.

**Behavior with/without explicitly knowing** $T$: Also, without explicitly knowing the value of $T$, the hierarchical schemes were able to infer the period, $T$, of the Environment by moving the first sub-list to the end of the list if two successive queries to the EOMA were not in the same group. These periodic variations were suffixed by "UnknownPeriod", yielding the MTF-MTF-EOMA-UnknownPeriod, MTF-TR-EOMA-UnknownPeriod, TR-MTF-EOMA-UnknownPeriod and TR-TR-EOMA-Unk-nownPeriod schemes. Figures 17 - 20 compare their relative performances against the regular hierarchical schemes that included the EOMA.

From Figure 17, we observe that the MTF-MTF-EOMA-Periodic boasted a superior performance to the MTF-MTF-EOMA for all sub-list variations except when $k = 64$ where the MTF-MTF-EOMA was marginally better. On the other hand, the MTF-MTF-EOMA-UnknownPeriod also outperformed the MTF-MTF-EOMA for all sub-list variations, and was on-par when $k = 64$. This observation is valid for the TR-MTF-EOMA-Periodic and UnknownPeriod setups, as shown in Figure 18.

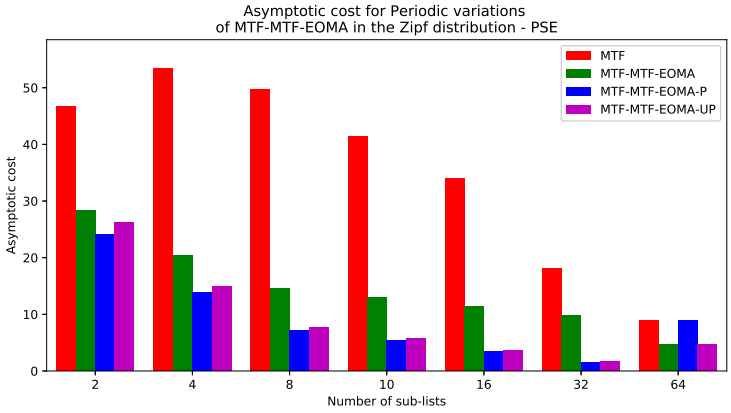



Figure 17: Asymptotic cost of Periodic variations of MTF-MTF-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

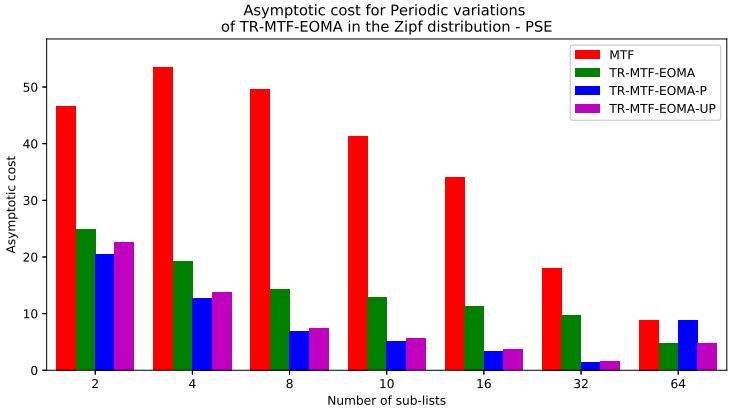



Figure 18: Asymptotic cost of Periodic variations of TR-MTF-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

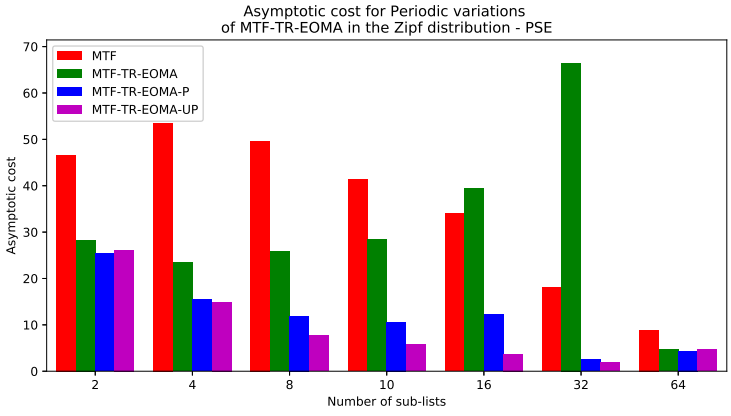



Figure 19: Asymptotic cost of Periodic variations of MTF-TR-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

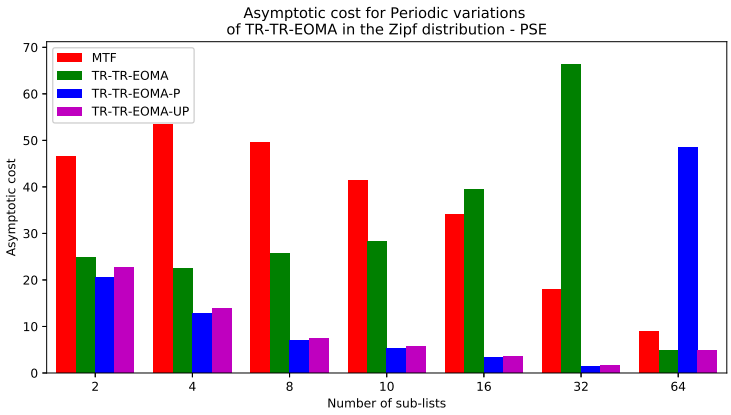



Figure 20: Asymptotic cost of Periodic variations of TR-TR-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

The MTF-TR-EOMA-Periodic scheme outperformed both the MTF and the MTF-TR-EOMA for all the sub-list variations. This result was more impressive as the "vanilla" MTF-TR-EOMA struggled against the MTF when the size of the sub-list, $k$, was greater than $16$. However, when compared to the MTF-TR-EOMA-Periodic, the asymptotic cost was categorically superior to the MTF and the MTF-TR-EOMA. The MTF-TR-EOMA-UnknownPeriod also outperformed the MTF for all sub-list variations. The TR-TR-EOMA-Periodic and UnknownPeriod, shown in Figure 20, had performances comparable to the MTF-TR-EOMA-Periodic/UnknownPeriod schemes. However, when $k = 64$, the TR-TR-EOMA-Periodic resulted in a poor performance that was not consistent with the rest of the results. This is possibly because when $k = 64$ and the list size is $128$, the number of elements within each sublist is $2$, where the MTF and TR are identical.

Although this section discusses the results for the Zipf distribution, the results for the 80-20, Lotka, Exponential and Linear distributions provided in the Appendix yielded similar patterns. They are omitted here in the interest of readability.

# 11 Conclusion

In this paper, we have considered the way by which data can be effectively stored, accessed and adaptively re-configured. This is an extremely pertinent issue with the advent of "Big Data" as a field. Unlike the Artificially Intelligence (AI)-based algorithms required, and the study of the hardware components to process the data, and the methods to efficiently store, access and re-configure the data has been relatively unexplored. This is precisely what we have studied in this paper. Indeed, these issues, which have been central to the field of Computer Science, are even more pertinent in these days when megabytes of data are being generated every second. This paper considers the problem of minimizing the cost of retrieval using the most fundamental data structure, i.e., a Singly-Linked List (SLL). We consider a SLL in which the elements are accessed by a *Non-stationary* Environment (NSE) exhibiting the so-called "Locality of Reference". We propose a solution to the problem by designing an "Adaptive" Data Structure (ADS), which is created by means of a composite of hierarchical data "sub"-structures to constitute the overall data structure. While the

primitive data structure in this paper is the SLL, these concepts can be easily extended to more complicated structures.

In this research, we studied the area of Adaptive Data Structures (ADSs). We considered the relatively novel concept of having lists whose basic primitive elements are themselves sub-lists, with ADS operations being done on the elements *and* on the sublists. To break the static arrangement of their sublists, we have incorporated the EOMA (from the field of Learning Automata (LA)) into the hierarchical schemes. The EOMA enabled the hierarchical schemes to capture the probabilistic dependence ordering of the query accesses from the Environment, and to achieve this in a deadlock-free manner. Further, the paper discussed the performance of the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and the TR-TR-EOMA for various sublist values of $k$, multiple distributions, and various types of non-stationarity.

The overall observation that we could make is that while the EOMA-augmented schemes are superior to the stand-alone MTF and TR schemes, the MTF-MTF-EOMA and the TR-MTF-EOMA perform better than the MTF-TR-EOMA and the TR-TR-EOMA. One can almost categorically state that, the schemes having the TR as its outer-list re-organization strategy were inferior to the MTF, when we compared their asymptotic and amortized costs. However, the observed poor performance of the MTF-TR-EOMA and the TR-TR-EOMA schemes as $k$ increases, were mitigated in the PSEs when a knowledge of the period, $T$, was incorporated into the hierarchical scheme.

A study of the various graphs that we have obtained seems to imply that there is a way by which we can group the various *schemes themselves* using a higher-level statistical analysis. Such a study remains open.

# References


[1] M. Agache and B. J. Oommen. Generalized pursuit learning schemes: new families of continuous and discretized learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(6):738–749, 2002.

[2] A. Amer. Adaptive list organizing strategies for non-stationary distributions. 2004.

[3] A. F. Atlasis, N. H. Loukas, and A. V. Vasilakos. The use of learning algorithms in atm networks call admission control problem: a methodology. *Computer Networks*, 34(3):341–353, 2000.

[4] A. F. Atlasis and A. V. Vasilakos. The use of reinforcement learning algorithms in traffic control of high speed networks. In *Advances in Computational Intelligence and Learning*, pages 353–369. Springer, 2002.

[5] R. Bachrach, R. El-Yaniv, and M. Reinstadtler. On the competitive theory and practice of online list accessing algorithms. *Algorithmica*, 32(2):201–245, 2002.

[6] A. J. Baddeley, R. Turner, et al. Spatstat: An r package for analyzing spatial point pattens, 2004.

[7] J. L. Bentley and C. C. McGeoch. Amortized analyses of self-organizing sequential search heuristics. *Communications of the ACM*, 28(4):404–411, 1985.

[8] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa. Stochastic properties of the random waypoint mobility model. *Wireless Networks*, 10(5):555–567, 2004.

[9] E. O. Bisong. On designing adaptive data structures with adaptive data "sub"-structures, MCS thesis, Carleton University, Ottawa, 2018.

[10] M. Fahimi and A. Ghasemi. A distributed learning automata scheme for spectrum management in self-organized cognitive radio network. *IEEE Transactions on Mobile Computing*, 16(6):1490–1501, 2016.

[11] E. Fayyoumi and B. J. Oommen. A fixed structure learning automaton micro-aggregation technique for secure statistical databases. In *International Conference on Privacy in Statistical Databases*, pages 114–128. Springer, 2006.

[12] E. Fayyoumi and B. J. Oommen. Achieving microaggregation for secure statistical databases using fixed-structure partitioning-based learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(5):1192–1205, 2009.

[13] W. Gale, S. Das, and C. T. Yu. Improvements to an algorithm for equipartitioning. *IEEE Transactions on Computers*, 39(5):706–710, 1990.

[14] J. H. Hester and D. S. Hirschberg. Self-organizing linear search. *ACM Computing Surveys (CSUR)*, 17(3):295–311, 1985.

[15] J. H. Hester and D. S. Hirschberg. Self-organizing search lists using probabilistic back-pointers. *Communications of the ACM*, 30(12):1074–1079, 1987.

[16] S. Irani. Two results on the list update problem. *Information Processing Letters*, 38(6):301–306, 1991.

[17] A. Jobava. Intelligent traffic-aware consolidation of virtual machines in a data center. Master's thesis, 2015.

[18] J. Kabudian, M. R. Meybodi, and M. M. Homayounpour. Applying continuous action reinforcement learning automata (carla) to global training of hidden markov models. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, volume 2, pages 638–642. IEEE, 2004.

[19] V. I. Krinsky. An asymptotically optimal automaton with exponential convergence. *Biofizika*, 9:484–87, 1964.

[20] V. Y. Krylov. One stochastic automaton which is asymptotically optimal in random medium. *Automation and Remote Control*, 24:1114–16, 1964.

[21] S Lakshmivarahan. Absolutely ezpedient learning algorithms for stochastic automata. *IEEE SMC*, 3(3):281–286, 1973.

[22] S. Lakshmivarahan. *Learning Algorithms Theory and Applications: Theory and Applications*. Springer Science & Business Media, 2012.

[23] J. K. Lanctot and B. J. Oommen. On discretizing estimator-based learning algorithms. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1417–1422. IEEE, 1991.

[24] J. K. Lanctôt and B. J. Oommen. Discretized estimator learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1473–1483, 1992.

[25] A. S. Mamaghani, M. Mahi, and M. R. Meybodi. A learning automaton based approach for data fragments allocation in distributed database systems. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 8–12. IEEE, 2010.

[26] M. R. Meybodi and H. Beigy. New learning automata based algorithms for adaptation of backpropagation algorithm parameters. *International Journal of Neural Systems*, 12(01):45–67, 2002.

[27] S. Misra and B J. Oommen. Gpspa: A new adaptive algorithm for maintaining shortest path routing trees in stochastic networks. *International Journal of Communication Systems*, 17(10):963–984, 2004.

[28] H. Mostafaei, A. Montieri, V. Persico, and A. Pescapé. A sleep scheduling approach based on learning automata for wsn partial coverage. *Journal of Network and Computer Applications*, 80:67–78, 2017.

[29] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Courier Corporation, 2012.

[30] M. S. Obaidat, G. I. Papadimitriou, A. S. Pomportsis, and H.S. Laskaridis. Learning automata-based bus arbitration for shared-medium atm switches. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(6):815–820, 2002.

[31] B. J. Oommen. Absorbing and ergodic discretized two-action learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(2):282–293, 1986.

[32] B J. Oommen and M. Agache. Continuous and discretized pursuit learning schemes: Various algorithms and their comparison. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(3):277–287, 2001.

[33] B. J. Oommen and C. Fothergill. The image examination and retrieval problem: A learning automaton-based solution. In *In Proceedings ICARCV92, International Conference on Automation, Robotics, and Computer Vision*. IEEE, 1992.

[34] B. J. Oommen and C. Fothergill. Fast learning automaton-based image examination and retrieval. *The Computer Journal*, 36(6):542–553, 1993.

[35] B. J. Oommen and J. K. Lanctôt. Discretized pursuit learning automata. *IEEE Transactions on systems, man, and cybernetics*, 20(4):931–938, 1990.

[36] B. J. Oommen and D. C. Y. Ma. Stochastic automata solutions to the object partitoning problem. *The Comput. Journal*, Vol. 35, A105A120, 1992.

[37] B. J. Oommen and D. C. Y. Ma. Deterministic learning automata solutions to the equipartitioning problem. *IEEE Transactions on Computers*, 37(1):2–14, 1988.

[38] B. J. Oommen and J. R. Zgierski. Breaking substitution cyphers using stochastic automata. *IEEE transactions on pattern analysis and machine intelligence*, 15(2):185–192, 1993.

[39] G. I. Papadimitriou and A. S. Pomportsis. Learning-automata-based tdma protocols for broadcast communication systems with bursty traffic. *IEEE Communications Letters*, 4(3):107–109, 2000.

[40] R. Rivest. On self-organizing sequential search heuristics. *Communications of the ACM*, 19(2):63–67, 1976.

[41] A. Shirvani. *Novel Solutions and Applications of the Object Partitioning Problem*. PhD thesis, Carleton University, Ottawa, 2018.

[42] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[43] A. M. Tenenbaum and R. M. Nemes. Two spectra of self-organizing sequential search algorithms. *SIAM Journal on Computing*, 11(3):557–566, 1982.

[44] M.A.L Thathachar and B J. Oommen. Discretized reward-inaction learning automata. *J. Cybernetics and Information Science*, 2:24–29, 1979.

[45] M.A.L. Thathachar and P. S Sastry. *Networks of learning automata: Techniques for online stochastic optimization*. Springer Science & Business Media, 2011.

[46] M. L. Tsetlin. Finite automata and models of simple forms of behaviour. *Russian mathematical surveys*, 18:1–27, 1963.

[47] F. M. Ung. Towards efficient and cost-effective live migrations of virtual machines. Master's thesis, 2015.

[48] A. Vasilakos, M. P. Saltouros, A.F. Atlassis, and W. Pedrycz. Optimizing qos routing in hierarchical atm networks using computational intelligence techniques. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 33(3):297–312, 2003.

[49] A. Yazidi, O. C. Granmo, and B. J. Oommen. Service selection in stochastic environments: a learning-automaton based solution. *Applied Intelligence*, 36(3):617–637, 2012.

[50] CT Yu, MK Siu, K Lam, and F Tai. Adaptive clustering schemes: General framework. In *Proc. of the IEEE COMPSAC Conference*, pages 81–89, 1981.

[51] X. Zhang, O. Granmo, and B J. Oommen. Discretized bayesian pursuit–a new scheme for reinforcement learning. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 784–793. Springer, 2012.

# Appendices

This Appendix contains additional experimental results for the EOMA Hierarchical SLLs-on-SLLs. The results contrast the performance of the MTF and TR with the hierarchical MTF-MTF-EOMA, TR-MTF-EOMA, MTF-TR-EOMA, and TR-TR-EOMA in dependent non-stationary Environments.

For a list containing 128 elements, the results are from an ensemble of 300,000 queries over 10 experiments. The ensemble asymptotic and amortized cost are used as evaluation metrics for the experiments. The asymptotic cost averages the access cost of the last 20% of the simulations, while the amortized cost is the average of the entire simulations access costs. The results are separated by the size of the sublist, $k = \{2, 4, 8, 16, 32, 64\}$, with the MSE utilizing values of $\alpha \in \{0.2, 0.9\}$, and for the PSE $T = \{30\}$.

| Strategy | Zipf | | 80-20 | | Lotka | | Exp | | Linear | |
|---|---|---|---|---|---|---|---|---|---|---|
| $k = 2$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 45.83 | 45.79 | 49.64 | 49.63 | 24.18 | 24.10 | 2.81 | 2.82 | 54.90 | 54.93 |
| TR | 37.43 | 37.83 | 41.51 | 41.87 | 19.01 | 19.39 | 2.48 | 2.65 | 48.36 | 48.76 |
| MTF-MTF-EOMA | 35.09 | 35.22 | 36.70 | 36.80 | 27.04 | 27.69 | 2.81 | 2.82 | 39.27 | 39.30 |
| MTF-TR-EOMA | 35.11 | 35.21 | 36.74 | 36.85 | 27.09 | 27.69 | 3.79 | 3.80 | 39.31 | 39.30 |
| TR-MTF-EOMA | 31.67 | 31.87 | 33.24 | 33.43 | 24.39 | 25.55 | 2.47 | 2.55 | 35.69 | 35.86 |
| TR-TR-EOMA | 31.60 | 31.86 | 33.21 | 33.45 | 24.42 | 26.06 | 2.48 | 2.55 | 35.76 | 35.90 |
| $k = 4$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 49.14 | 49.19 | 50.79 | 50.78 | 38.11 | 38.24 | 4.81 | 4.83 | 52.01 | 51.98 |
| TR | 48.69 | 49.11 | 50.90 | 51.27 | 36.56 | 36.87 | 5.04 | 5.33 | 53.29 | 53.74 |
| MTF-MTF-EOMA | 25.62 | 25.81 | 26.05 | 26.14 | 23.56 | 25.83 | 6.63 | 6.80 | 26.39 | 26.47 |
| MTF-TR-EOMA | 28.73 | 28.82 | 28.99 | 29.11 | 26.55 | 27.18 | 6.59 | 6.84 | 29.23 | 29.40 |
| TR-MTF-EOMA | 24.56 | 24.70 | 24.97 | 25.12 | 22.18 | 23.25 | 6.51 | 7.31 | 25.27 | 25.43 |
| TR-TR-EOMA | 27.51 | 27.68 | 27.97 | 28.18 | 25.17 | 26.35 | 6.76 | 7.51 | 28.35 | 28.49 |
| $k = 8$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 43.35 | 43.25 | 43.76 | 43.82 | 39.30 | 39.17 | 8.72 | 8.71 | 43.60 | 43.64 |
| TR | 55.44 | 55.85 | 56.74 | 56.96 | 48.25 | 48.66 | 10.52 | 10.93 | 56.79 | 57.26 |
| MTF-MTF-EOMA | 19.14 | 19.35 | 19.23 | 19.40 | 18.70 | 19.26 | 12.34 | 12.90 | 19.31 | 19.45 |
| MTF-TR-EOMA | 27.80 | 27.93 | 27.77 | 28.02 | 27.17 | 27.54 | 16.89 | 16.57 | 28.04 | 28.08 |
| TR-MTF-EOMA | 18.84 | 19.09 | 18.99 | 19.18 | 18.37 | 19.07 | 12.87 | 13.35 | 18.96 | 19.18 |
| TR-TR-EOMA | 27.55 | 27.72 | 27.62 | 27.80 | 26.96 | 27.25 | 17.17 | 17.10 | 27.70 | 27.87 |
| $k = 16$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 32.22 | 32.33 | 32.54 | 32.51 | 31.17 | 31.27 | 15.38 | 15.46 | 32.11 | 32.14 |
| TR | 59.01 | 59.22 | 59.74 | 59.85 | 55.24 | 55.40 | 21.45 | 21.96 | 59.04 | 59.31 |
| MTF-MTF-EOMA | 15.49 | 15.82 | 15.51 | 15.87 | 15.34 | 15.77 | 18.63 | 19.03 | 15.45 | 15.88 |
| MTF-TR-EOMA | 32.73 | 32.95 | 32.88 | 32.92 | 32.59 | 32.75 | 28.31 | 28.07 | 32.93 | 32.90 |
| TR-MTF-EOMA | 15.41 | 15.75 | 15.53 | 15.83 | 15.40 | 15.83 | 18.85 | 19.59 | 15.53 | 15.88 |
| TR-TR-EOMA | 32.74 | 32.90 | 32.66 | 32.79 | 32.68 | 32.80 | 28.47 | 28.45 | 32.51 | 32.92 |
| $k = 32$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 20.93 | 20.94 | 20.93 | 20.95 | 20.74 | 20.77 | 17.52 | 17.60 | 20.74 | 20.72 |
| TR | 60.36 | 60.33 | 60.40 | 60.44 | 58.17 | 58.39 | 36.66 | 37.06 | 59.87 | 59.76 |
| MTF-MTF-EOMA | 18.27 | 19.65 | 16.28 | 18.28 | 19.62 | 20.63 | 22.69 | 22.81 | 17.31 | 18.87 |
| MTF-TR-EOMA | 40.80 | 40.86 | 41.27 | 40.97 | 40.81 | 40.58 | 32.22 | 32.20 | 40.88 | 40.73 |
| TR-MTF-EOMA | 18.07 | 19.37 | 18.69 | 19.92 | 18.44 | 20.25 | 22.74 | 22.69 | 16.43 | 19.15 |
| TR-TR-EOMA | 41.01 | 41.03 | 41.44 | 40.83 | 40.99 | 40.68 | 32.64 | 32.29 | 41.04 | 40.78 |
| $k = 64$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 12.02 | 12.10 | 12.12 | 12.11 | 12.07 | 12.07 | 11.73 | 11.78 | 11.91 | 11.95 |
| TR | 58.93 | 58.97 | 59.17 | 59.15 | 58.51 | 58.38 | 48.91 | 49.13 | 58.29 | 58.28 |
| MTF-MTF-EOMA | 12.10 | 12.12 | 12.12 | 12.10 | 12.10 | 12.08 | 11.83 | 11.87 | 12.11 | 12.14 |
| MTF-TR-EOMA | 12.14 | 12.13 | 12.14 | 12.14 | 12.06 | 12.11 | 11.92 | 11.86 | 11.99 | 12.12 |
| TR-MTF-EOMA | 12.19 | 12.13 | 12.12 | 12.09 | 12.05 | 12.08 | 11.88 | 11.87 | 12.11 | 12.12 |
| TR-TR-EOMA | 12.17 | 12.15 | 12.07 | 12.13 | 12.06 | 12.09 | 11.85 | 11.87 | 12.08 | 12.12 |

Table 9: EOMA-Augmented Hierarchical SLLs Results for MSE with $\alpha = 0.9$ and $k = 2, 4, 8, 16, 32, 64$.

| Strategy | Zipf | | 80-20 | | Lotka | | Exp | | Linear | |
|---|---|---|---|---|---|---|---|---|---|---|
| $k = 2$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 47.27 | 47.17 | 51.19 | 51.18 | 24.64 | 24.67 | 3.72 | 3.73 | 56.80 | 56.83 |
| TR | 37.51 | 37.93 | 41.38 | 41.81 | 19.02 | 19.40 | 2.82 | 2.99 | 48.30 | 48.70 |
| MTF-MTF-EOMA | 65.00 | 65.04 | 64.68 | 64.96 | 55.57 | 54.89 | 3.71 | 3.73 | 63.04 | 63.31 |
| MTF-TR-EOMA | 64.54 | 65.03 | 64.83 | 64.81 | 55.09 | 54.68 | 3.72 | 3.73 | 63.63 | 63.48 |
| TR-MTF-EOMA | 70.02 | 70.24 | 68.88 | 68.93 | 71.55 | 71.53 | 2.81 | 2.88 | 66.66 | 66.76 |
| TR-TR-EOMA | 70.21 | 70.18 | 68.89 | 68.86 | 71.94 | 71.39 | 2.80 | 2.88 | 66.64 | 66.76 |
| $k = 4$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 56.23 | 56.21 | 58.05 | 58.03 | 43.70 | 43.66 | 7.72 | 7.71 | 59.80 | 59.77 |
| TR | 48.87 | 49.18 | 51.04 | 51.40 | 36.39 | 36.84 | 5.77 | 6.03 | 53.40 | 53.79 |
| MTF-MTF-EOMA | 62.61 | 62.76 | 63.17 | 63.22 | 57.31 | 57.41 | 30.65 | 29.62 | 63.39 | 63.39 |
| MTF-TR-EOMA | 62.67 | 62.76 | 63.05 | 63.11 | 57.26 | 57.33 | 30.16 | 30.84 | 63.33 | 63.36 |
| TR-MTF-EOMA | 63.61 | 63.68 | 63.87 | 63.84 | 60.45 | 60.44 | 38.47 | 39.74 | 63.83 | 63.87 |
| TR-TR-EOMA | 63.76 | 63.68 | 63.81 | 63.85 | 60.33 | 60.27 | 39.87 | 39.22 | 63.92 | 63.87 |
| $k = 8$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 60.16 | 60.18 | 60.99 | 60.98 | 54.36 | 54.33 | 15.49 | 15.47 | 61.08 | 61.08 |
| TR | 55.71 | 56.05 | 56.89 | 57.21 | 48.47 | 48.89 | 11.64 | 12.05 | 57.18 | 57.51 |
| MTF-MTF-EOMA | 62.80 | 62.79 | 62.95 | 63.01 | 61.03 | 60.98 | 38.57 | 38.57 | 63.22 | 63.24 |
| MTF-TR-EOMA | 62.97 | 62.91 | 63.03 | 63.08 | 61.16 | 61.20 | 38.54 | 38.45 | 63.26 | 63.25 |
| TR-MTF-EOMA | 62.91 | 62.89 | 63.10 | 63.05 | 61.23 | 61.26 | 40.37 | 40.55 | 63.23 | 63.29 |
| TR-TR-EOMA | 62.95 | 62.95 | 63.14 | 63.13 | 61.42 | 61.47 | 40.22 | 40.32 | 63.27 | 63.29 |
| $k = 16$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 61.13 | 61.17 | 61.38 | 61.45 | 58.79 | 58.81 | 30.10 | 30.05 | 60.98 | 61.01 |
| TR | 59.62 | 59.95 | 60.31 | 60.52 | 55.64 | 56.04 | 22.98 | 23.44 | 59.63 | 59.99 |
| MTF-MTF-EOMA | 63.16 | 63.08 | 63.22 | 63.15 | 62.55 | 62.56 | 50.71 | 50.71 | 63.09 | 63.05 |
| MTF-TR-EOMA | 63.43 | 63.41 | 63.37 | 63.42 | 63.24 | 63.21 | 53.11 | 53.12 | 63.16 | 63.19 |
| TR-MTF-EOMA | 63.09 | 63.13 | 63.11 | 63.16 | 62.59 | 62.63 | 50.90 | 50.92 | 63.10 | 63.11 |
| TR-TR-EOMA | 63.42 | 63.44 | 63.56 | 63.47 | 63.33 | 63.25 | 53.45 | 53.36 | 63.15 | 63.17 |
| $k = 32$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 60.02 | 60.06 | 60.12 | 60.15 | 59.25 | 59.27 | 46.83 | 46.84 | 59.53 | 59.60 |
| TR | 61.78 | 61.93 | 61.94 | 62.13 | 59.61 | 59.94 | 38.88 | 39.30 | 61.19 | 61.43 |
| MTF-MTF-EOMA | 62.77 | 62.76 | 62.74 | 62.78 | 62.63 | 62.63 | 58.08 | 58.16 | 62.66 | 62.57 |
| MTF-TR-EOMA | 63.35 | 63.41 | 63.47 | 63.46 | 63.48 | 63.42 | 62.60 | 62.51 | 63.12 | 63.27 |
| TR-MTF-EOMA | 62.79 | 62.76 | 62.77 | 62.84 | 62.69 | 62.62 | 58.11 | 58.18 | 62.55 | 62.57 |
| TR-TR-EOMA | 63.52 | 63.40 | 63.45 | 63.40 | 63.34 | 63.39 | 62.53 | 62.49 | 63.25 | 63.24 |
| $k = 64$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 56.89 | 56.94 | 56.89 | 56.90 | 56.65 | 56.72 | 53.84 | 53.82 | 56.34 | 56.33 |
| TR | 62.68 | 62.81 | 62.73 | 62.89 | 61.70 | 61.87 | 52.75 | 53.07 | 61.93 | 62.00 |
| MTF-MTF-EOMA | 61.26 | 61.32 | 61.35 | 61.32 | 61.09 | 61.19 | 60.56 | 60.57 | 61.12 | 61.19 |
| MTF-TR-EOMA | 63.28 | 63.35 | 63.33 | 63.35 | 63.30 | 63.32 | 63.27 | 63.31 | 63.26 | 63.34 |
| TR-MTF-EOMA | 61.26 | 61.29 | 61.25 | 61.28 | 61.28 | 61.27 | 60.59 | 60.54 | 61.22 | 61.21 |
| TR-TR-EOMA | 63.36 | 63.31 | 63.38 | 63.38 | 63.37 | 63.35 | 63.30 | 63.31 | 63.39 | 63.38 |

Table 10: EOMA-Augmented Hierarchical SLLs Results for MSE with $\alpha = 0.2$ and $k = 2, 4, 8, 16, 32, 64$.

| Strategy | Zipf | | 80-20 | | Lotka | | Exp | | Linear | |
|---|---|---|---|---|---|---|---|---|---|---|
| $k = 2$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 46.61 | 46.62 | 50.66 | 50.57 | 24.27 | 24.23 | 2.52 | 2.53 | 56.21 | 56.25 |
| TR | 37.49 | 37.92 | 41.35 | 41.88 | 19.02 | 19.42 | 2.45 | 2.62 | 48.27 | 48.75 |
| MTF-MTF-EOMA | 28.33 | 28.40 | 29.96 | 30.04 | 20.33 | 20.55 | 2.84 | 2.85 | 32.48 | 32.47 |
| MTF-TR-EOMA | 28.28 | 28.39 | 29.98 | 30.07 | 20.35 | 20.61 | 2.51 | 2.52 | 32.44 | 32.49 |
| TR-MTF-EOMA | 24.80 | 25.04 | 26.41 | 26.59 | 17.66 | 18.28 | 2.43 | 2.50 | 28.94 | 29.07 |
| TR-TR-EOMA | 24.84 | 25.07 | 26.42 | 26.61 | 17.68 | 18.20 | 2.43 | 2.50 | 28.97 | 29.11 |
| MTF-MTF-EOMA-P | 24.09 | 24.18 | 25.68 | 25.76 | 16.05 | 16.50 | 4.65 | 4.66 | 28.21 | 28.26 |
| MTF-MTF-EOMA-UP | 26.21 | 26.29 | 27.81 | 27.92 | 18.20 | 18.65 | 2.51 | 2.52 | 30.33 | 30.37 |
| MTF-TR-EOMA-P | 24.03 | 24.19 | 25.73 | 25.79 | 16.05 | 16.33 | 4.66 | 4.66 | 28.17 | 28.25 |
| MTF-TR-EOMA-UP | 26.17 | 26.32 | 27.85 | 27.93 | 18.16 | 18.54 | 2.52 | 2.52 | 30.32 | 30.37 |
| TR-MTF-EOMA-P | 20.54 | 20.77 | 22.20 | 22.38 | 13.39 | 13.98 | 4.57 | 4.63 | 24.68 | 24.83 |
| TR-MTF-EOMA-UP | 22.67 | 22.92 | 24.30 | 24.46 | 15.55 | 16.35 | 2.51 | 2.59 | 26.81 | 26.98 |
| TR-TR-EOMA-P | 20.56 | 20.80 | 22.14 | 22.35 | 13.39 | 14.13 | 4.23 | 4.30 | 24.70 | 24.82 |
| TR-TR-EOMA-UP | 22.65 | 22.92 | 24.29 | 24.44 | 15.56 | 15.96 | 2.42 | 2.50 | 26.84 | 26.98 |
| $k = 4$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 53.48 | 53.48 | 55.28 | 55.34 | 41.19 | 41.21 | 4.49 | 4.50 | 56.87 | 56.93 |
| TR | 48.72 | 49.16 | 51.10 | 51.41 | 36.51 | 36.93 | 5.37 | 5.60 | 53.44 | 53.82 |
| MTF-MTF-EOMA | 20.26 | 20.37 | 20.65 | 20.75 | 18.17 | 18.41 | 3.73 | 3.98 | 20.99 | 21.07 |
| MTF-TR-EOMA | 23.48 | 23.59 | 23.82 | 23.91 | 21.37 | 21.58 | 4.12 | 4.21 | 24.18 | 24.27 |
| TR-MTF-EOMA | 19.19 | 19.29 | 19.59 | 19.71 | 16.93 | 17.27 | 4.36 | 4.23 | 19.96 | 20.05 |
| TR-TR-EOMA | 22.36 | 22.49 | 22.80 | 22.88 | 20.12 | 20.46 | 4.54 | 4.46 | 23.16 | 23.26 |
| MTF-MTF-EOMA-P | 13.89 | 13.95 | 14.24 | 14.32 | 11.75 | 11.98 | 6.30 | 6.46 | 14.60 | 14.67 |
| MTF-MTF-EOMA-UP | 14.96 | 15.06 | 15.29 | 15.42 | 12.85 | 13.15 | 5.35 | 5.63 | 15.66 | 15.78 |
| MTF-TR-EOMA-P | 13.87 | 13.96 | 14.27 | 14.33 | 11.79 | 12.02 | 8.77 | 9.04 | 14.60 | 14.69 |
| MTF-TR-EOMA-UP | 14.95 | 15.06 | 15.31 | 15.43 | 12.82 | 13.09 | 6.47 | 6.18 | 15.65 | 15.78 |
| TR-MTF-EOMA-P | 12.77 | 12.89 | 13.21 | 13.31 | 10.52 | 10.86 | 6.49 | 6.79 | 13.57 | 13.69 |
| TR-MTF-EOMA-UP | 13.86 | 14.00 | 14.25 | 14.40 | 11.61 | 11.96 | 5.93 | 5.73 | 14.64 | 14.77 |
| TR-TR-EOMA-P | 12.77 | 12.89 | 13.20 | 13.32 | 10.53 | 10.83 | 9.31 | 9.10 | 13.57 | 13.67 |
| TR-TR-EOMA-UP | 13.83 | 14.02 | 14.30 | 14.40 | 11.59 | 11.95 | 6.14 | 6.03 | 14.65 | 14.77 |
| $k = 8$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 49.64 | 49.62 | 50.24 | 50.23 | 44.52 | 44.53 | 8.46 | 8.48 | 50.08 | 50.06 |
| TR | 55.65 | 56.09 | 56.91 | 57.28 | 48.51 | 48.91 | 11.18 | 11.58 | 57.19 | 57.60 |
| MTF-MTF-EOMA | 14.63 | 14.76 | 14.70 | 14.84 | 14.12 | 14.31 | 8.59 | 8.68 | 14.72 | 14.84 |
| MTF-TR-EOMA | 25.82 | 25.93 | 25.90 | 26.01 | 25.32 | 25.44 | 13.88 | 12.49 | 25.92 | 26.02 |
| TR-MTF-EOMA | 14.39 | 14.54 | 14.49 | 14.62 | 13.76 | 14.03 | 8.92 | 9.69 | 14.50 | 14.63 |
| TR-TR-EOMA | 25.58 | 25.71 | 25.69 | 25.80 | 24.97 | 25.12 | 13.70 | 13.11 | 25.70 | 25.80 |
| MTF-MTF-EOMA-P | 7.16 | 7.28 | 7.24 | 7.38 | 6.66 | 6.82 | 6.14 | 7.53 | 7.26 | 7.40 |
| MTF-MTF-EOMA-UP | 7.69 | 7.86 | 7.78 | 7.95 | 7.19 | 7.48 | 8.90 | 10.57 | 7.79 | 7.92 |
| MTF-TR-EOMA-P | 7.16 | 7.30 | 7.24 | 7.36 | 6.66 | 6.84 | 9.21 | 12.55 | 7.25 | 7.38 |
| MTF-TR-EOMA-UP | 7.69 | 7.92 | 7.77 | 7.96 | 7.19 | 7.52 | 8.79 | 11.62 | 7.78 | 7.99 |
| TR-MTF-EOMA-P | 6.91 | 7.05 | 7.02 | 7.16 | 6.29 | 6.50 | 6.30 | 8.06 | 7.03 | 7.17 |
| TR-MTF-EOMA-UP | 7.45 | 7.62 | 7.53 | 7.72 | 6.83 | 7.11 | 9.51 | 11.91 | 7.56 | 7.73 |
| TR-TR-EOMA-P | 6.91 | 7.05 | 7.01 | 7.16 | 6.30 | 6.53 | 11.05 | 13.05 | 7.04 | 7.20 |
| TR-TR-EOMA-UP | 7.45 | 7.67 | 7.55 | 7.76 | 6.83 | 7.13 | 9.47 | 12.65 | 7.56 | 7.75 |

Table 11: EOMA-Augmented Hierarchical SLLs Results for PSE with $T = 30$ and $k = 2, 4, 8$.

| Strategy | Zipf | | 80-20 | | Lotka | | Exp | | Linear | |
|---|---|---|---|---|---|---|---|---|---|---|
| $k = 16$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 34.00 | 33.95 | 34.04 | 34.05 | 32.70 | 32.68 | 15.32 | 15.32 | 33.58 | 33.55 |
| TR | 59.73 | 59.99 | 60.25 | 60.59 | 55.72 | 56.05 | 23.00 | 23.40 | 59.78 | 60.08 |
| MTF-MTF-EOMA | 11.44 | 11.64 | 11.45 | 11.65 | 11.57 | 11.32 | 12.29 | 13.38 | 11.42 | 11.64 |
| MTF-TR-EOMA | 39.43 | 39.47 | 39.45 | 39.48 | 39.32 | 39.25 | 30.31 | 27.71 | 39.42 | 39.46 |
| TR-MTF-EOMA | 11.40 | 11.60 | 11.41 | 11.65 | 11.26 | 11.58 | 13.52 | 14.39 | 11.39 | 11.59 |
| TR-TR-EOMA | 39.39 | 39.43 | 39.41 | 39.44 | 39.26 | 39.26 | 28.72 | 28.18 | 39.39 | 39.42 |
| MTF-MTF-EOMA-P | 3.43 | 3.64 | 3.45 | 3.67 | 3.32 | 3.56 | 5.99 | 7.99 | 3.42 | 3.61 |
| MTF-MTF-EOMA-UP | 3.70 | 3.97 | 3.71 | 3.99 | 3.58 | 3.96 | 10.44 | 14.19 | 3.69 | 3.99 |
| MTF-TR-EOMA-P | 3.43 | 3.75 | 3.45 | 3.74 | 3.32 | 3.70 | 11.00 | 15.43 | 3.43 | 3.73 |
| MTF-TR-EOMA-UP | 3.70 | 4.04 | 3.72 | 4.12 | 3.58 | 4.16 | 25.53 | 25.70 | 3.69 | 4.03 |
| TR-MTF-EOMA-P | 3.40 | 3.61 | 3.41 | 3.61 | 3.26 | 3.49 | 6.33 | 7.83 | 3.39 | 3.60 |
| TR-MTF-EOMA-UP | 3.66 | 3.97 | 3.67 | 3.95 | 3.52 | 3.90 | 14.71 | 16.67 | 3.66 | 3.97 |
| TR-TR-EOMA-P | 3.39 | 3.71 | 3.41 | 3.71 | 3.26 | 3.65 | 11.73 | 16.20 | 3.39 | 3.66 |
| TR-TR-EOMA-UP | 3.66 | 4.01 | 3.68 | 4.06 | 3.53 | 4.00 | 21.10 | 24.96 | 3.65 | 4.05 |
| $k = 32$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 18.01 | 17.99 | 18.01 | 18.00 | 17.89 | 17.89 | 16.45 | 16.46 | 17.74 | 17.73 |
| TR | 61.83 | 62.01 | 62.14 | 62.31 | 59.78 | 62.11 | 39.03 | 39.40 | 61.36 | 61.61 |
| MTF-MTF-EOMA | 9.75 | 11.01 | 9.75 | 11.06 | 10.98 | 9.73 | 15.54 | 16.79 | 9.80 | 11.02 |
| MTF-TR-EOMA | 66.33 | 62.47 | 66.32 | 62.43 | 66.30 | 61.95 | 22.28 | 21.29 | 66.20 | 61.96 |
| TR-MTF-EOMA | 10.90 | 9.75 | 9.75 | 10.97 | 9.72 | 11.16 | 15.48 | 16.57 | 9.74 | 10.87 |
| TR-TR-EOMA | 66.35 | 62.28 | 66.14 | 62.10 | 66.17 | 61.96 | 22.64 | 21.64 | 66.23 | 62.21 |
| MTF-MTF-EOMA-P | 1.49 | 2.84 | 1.49 | 2.80 | 1.46 | 2.86 | 13.97 | 14.74 | 1.48 | 2.81 |
| MTF-MTF-EOMA-UP | 1.62 | 4.02 | 1.62 | 3.75 | 1.60 | 4.37 | 18.85 | 19.28 | 1.60 | 3.98 |
| MTF-TR-EOMA-P | 1.48 | 6.01 | 1.49 | 6.65 | 1.46 | 7.82 | 42.38 | 43.96 | 1.47 | 6.11 |
| MTF-TR-EOMA-UP | 1.62 | 5.93 | 1.62 | 5.52 | 1.60 | 6.52 | 30.40 | 28.87 | 1.61 | 5.64 |
| TR-MTF-EOMA-P | 1.48 | 2.81 | 1.48 | 2.82 | 1.46 | 3.03 | 12.82 | 13.91 | 1.47 | 2.76 |
| TR-MTF-EOMA-UP | 1.61 | 3.79 | 1.62 | 3.55 | 1.59 | 4.46 | 18.74 | 19.16 | 1.61 | 3.87 |
| TR-TR-EOMA-P | 1.48 | 6.39 | 1.48 | 6.38 | 1.45 | 7.47 | 41.70 | 44.10 | 1.47 | 6.53 |
| TR-TR-EOMA-UP | 1.61 | 6.04 | 1.62 | 6.46 | 1.59 | 6.55 | 30.66 | 29.46 | 1.60 | 6.64 |
| $k = 64$ | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor | Asym | Amor |
| MTF | 8.89 | 8.88 | 8.89 | 8.88 | 8.89 | 8.88 | 8.84 | 8.83 | 8.76 | 8.74 |
| TR | 62.94 | 63.05 | 63.08 | 63.15 | 62.04 | 62.22 | 53.05 | 53.28 | 62.20 | 62.27 |
| MTF-MTF-EOMA | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |
| MTF-TR-EOMA | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |
| TR-MTF-EOMA | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |
| TR-TR-EOMA | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |
| MTF-MTF-EOMA-P | 8.83 | 8.83 | 8.83 | 8.83 | 8.82 | 8.82 | 8.77 | 8.77 | 8.82 | 8.82 |
| MTF-MTF-EOMA-UP | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |
| MTF-TR-EOMA-P | 48.54 | 48.56 | 48.54 | 48.56 | 48.54 | 48.56 | 48.48 | 48.51 | 48.51 | 48.54 |
| MTF-TR-EOMA-UP | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |
| TR-MTF-EOMA-P | 8.83 | 8.83 | 8.83 | 8.83 | 8.82 | 8.82 | 8.78 | 8.78 | 8.82 | 8.82 |
| TR-MTF-EOMA-UP | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |
| TR-TR-EOMA-P | 48.54 | 48.56 | 48.54 | 48.56 | 48.54 | 48.56 | 48.48 | 48.50 | 48.51 | 48.54 |
| TR-TR-EOMA-UP | 4.73 | 4.73 | 4.73 | 4.73 | 4.72 | 4.72 | 4.67 | 4.67 | 4.72 | 4.72 |

Table 12: EOMA-Augmented Hierarchical SLLs Results for PSE with $T = 30$ and $k = 16, 32, 64$.