On Designing Adaptive Data Structures with Adaptive Data "Sub"-Structures

By
Ekaba Bisong

A thesis proposal submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfilment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

October 2018

The undersigned hereby recommend to
the Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis,

On Designing Adaptive Data Structures

with Adaptive Data "Sub"-Structures

submitted by

Ekaba Bisong

Dr. Douglas Howe
(Director, School of Computer Science)

Dr. B. John Oommen
(Thesis Supervisor)

Carleton University

October 2018

Abstract

Data structures are key pillars for optimizing computational efficiency, as they contribute in no small measure to enhancing the "speed" in the accessing and subsequent processing of data. The need for enhancing speed is critical for almost all applications and domains, and this is most relevant when real-time or near real-time efficiency is desired.

This thesis proposes the use of "Adaptive" Data-Structures (ADSs) that invoke reinforcement learning schemes from the theory of Learning Automata (LA). These operate in conjunction with select re-organization rules to update themselves as they receive queries from the Environment of interaction. The result of such a process is the subsequent minimization of the cost associated with query accesses. The Environments under consideration are those that exhibit a so-called "locality of reference", and are referred to as Non-stationary Environments (NSEs).

A hierarchy of data "sub"-structures is used to design Singly-Linked Lists (SLLs) on Singly-Linked Lists, which thus contain outer and sub-list contexts. The elements that are more likely to be accessed *together* are grouped within the same sub-context, while the sub-lists themselves are moved "en masse" towards the head of the list-context by following a re-organization strategy.

The Object Migration Automaton (OMA) family of reinforcement schemes are employed to capture the probabilistic dependence of the elements in the data structure as it receives query accesses from the Environment. The Enhanced Object Migration Automaton (EOMA), the *Pursuit* Enhanced Object Migration Automaton (PEOMA), and the *Transitivity* Pursuit Enhanced Object Migration Automaton (TPEOMA) have each been individually incorporated into the hierarchical SLLs. The consequent results are currently the state-of-the-art methods for adaptive SLLs operating in NSEs.

Acknowledgements

First and foremost, I give thanks to the Lord, my God, for He is good, and His mercies endure forever. Let all the earth praise the Name of the Lord God and His Son Jesus Christ, who is blessed forever and ever.

I am particularly grateful to my Supervisor Prof. B. John Oommen. He has been a rock and a support to me. He taught me all that I know about the field of learning automata, and by extension, the broader body of reinforcement learning. Prof. Oommen was my mentor and received me as his son. He was an example of what it means to live in holiness and righteousness. I am privileged to sit and learn under him.

I would like to offer special thanks to my parents, Prof. Francis and Nonso Bisong. They have supported me at all stages of my life and education. My parents gave me an indescribable gift by teaching me the way of the Lord, and this has contributed to my peace. My father taught me the virtues of hard-work and the disciplines of being a godly man. He is my role-model, and an example of the man I hope to grow into. My mother has been my best-friend and my confidant. She has been there to support and advise me, and to comfort me in the most difficult moments of my life. I am tremendously indebted to my mother.

My siblings Osowo-Ayim, Chidera and Ginika have been my friends, and they have made the burden of this journey lighter. I could not have made it without their love, support and help. I will also like to thank my dear friend Rasine Ukene for always supporting and believing in me, and my room-mates Jonathan and Christina Austin for their friendship and help. Christina assisted me in the final days to prepare the Appendix of this work.

Finally, I will want to especially thank my Uncle and Aunt, Achu and Blessing Bisong, for their love, care, support and encouragement throughout the time of this thesis, and indeed, my stay in Canada. They have been a rock in my life. I am grateful to the faculty, the staff and the friends I made at the School of Computer Science at Carleton. They made my stay here enjoyable and fulfilling.

Contents

1 Introduction **1**

1.1 Introduction 1
1.2 Motivation of the Thesis 3
1.3 Problem Statement of the Thesis 4
1.4 Objectives of the Thesis 6
1.5 Contributions of the Thesis 7
1.6 Hardware/Software Used for the Simulations 8
1.7 Organization of the Thesis 8

2 Background Survey **10**

2.1 Introduction 10
2.2 Learning Automata 12
2.2.1 Fixed Structure Stochastic Automaton 16
2.2.2 Variable Structure Stochastic Automaton 20
2.2.3 Estimator and Pursuit Schemes 23
2.2.4 Non-stationary Environments 24
2.3 Strategies for Solving the Object Partitioning Problem 26
2.3.1 The Case for Partitioning 27
2.3.2 The Basic Adaptive Method (BAM) 30
2.3.3 Tsetlin and Krinsky Strategies to the OPP 32
2.3.4 The Object Migration Automaton (OMA) 35
2.3.5 The Stochastic Move Automaton (SMA) 41

2.3.6 The Modified Linear Reward Penalty Scheme (ML_{RP}) 42
2.3.7 The *Pursuit*-OMA (POMA) 45
2.3.8 Enhanced-OMA (EOMA) / *Pursuit*-EOMA (PEOMA) 48
2.3.9 Transitivity for PEOMA - (TPEOMA) 51
2.4 Adaptive Lists 54
2.4.1 Performance Evaluation 59
2.4.2 Convergence 65
2.4.3 Deterministic Adaptive Approaches 65
2.4.4 Move-To-Front (MTF) 66
2.4.5 Transposition Rule (TR) 67
2.4.6 Composite MTF & TR Strategies 70
2.4.7 Other Deterministic Adaptive Strategies 71
2.4.8 Probabilistic Approaches 75
2.4.9 Concluding Remarks 76

3 Adaptive Data "Sub"-Structures 78
3.1 Introduction 78
3.2 Locality of Reference 79
3.2.1 Markovian Switching Environment (MSE) 79
3.2.2 Periodic Switching Environment (PSE) 80
3.3 Generating Query Distributions 81
3.4 The Rationale for using Data "sub"-structures 83
3.4.1 Designing the Hierarchical Schemes 84
3.4.2 Challenges of the Hierarchical Schemes 85
3.5 Hierarchical List Reorganization Strategies 85
3.5.1 MTF-MTF 86
3.5.2 MTF-TR 87
3.5.3 TR-MTF 88
3.5.4 TR-TR 89
3.6 Concluding Remarks 91

4 EOMA-Augmented Hierarchical SLLs **94**
4.1 Introduction 94
4.2 The EOMA *vs* the OMA 95
4.3 MTF-MTF-EOMA 96
4.4 MTF-TR-EOMA 99
4.5 TR-MTF-EOMA 99
4.6 TR-TR-EOMA 101
4.7 Performance in MSEs 101
4.8 Performance in PSEs 111
4.9 Results for Periodic Variations 117
4.10 Concluding Remarks 121

5 PEOMA/TPEOMA-Augmented Hierarchical SLLs **122**
5.1 Introduction 122
5.2 MTF-MTF-PEOMA 123
5.3 MTF-TR-PEOMA 123
5.4 TR-MTF-PEOMA 124
5.5 TR-TR-PEOMA 125
5.6 Performance of PEOMA-Hierarchical Schemes in MSEs 126
5.7 Performance of PEOMA-Hierarchical Schemes in PSEs 132
5.8 Results for Periodic variations with PEOMA-Hierarchical Schemes . . 138
5.9 MTF-MTF-TPEOMA 138
5.10 MTF-TR-TPEOMA 140
5.11 TR-MTF-TPEOMA 141
5.12 TR-TR-TPEOMA 142
5.13 Performance of TPEOMA-Hierarchical Schemes in MSE 142
5.14 Performance of TPEOMA-Hierarchical Schemes in PSEs 144
5.15 Results for Periodic variations with TPEOMA-Hierarchical Schemes . 152
5.16 Concluding Remarks 154

6 Summary and Conclusion **157**
6.1 Summary of the Thesis 157

6.2 Conclusion of the Thesis 160
6.3 Publications from the Thesis 161
6.4 Future Work 161

Bibliography **163**

Appendices **170**

A OMA-Augmented Hierarchical SLLs Results **171**

B EOMA-Augmented Hierarchical SLLs Results **176**

C PEOMA-Augmented Hierarchical SLLs Results **181**

D TPEOMA-Augmented Hierarchical SLLs Results **186**

List of Figures

2.1 The Learning Automata Model 13
2.2 The transition map of the $L_{2N,2}$ Automaton 17
2.3 The transition map of Krinsky's Automaton 19
2.4 The transition map of Krylov's Automaton for receiving a penalty . . 19
2.5 The model for a PSE in the case of FSSA. 26
2.6 The model for a PSE in the case of VSSA. 27
2.7 The Partitioning Algorithm 30
2.8 The Multi-action transition graph for the Tsetlin L_{KN}, K automaton ($K = 4$). 33
2.9 The OMA Algorithm 38
2.10 The Markov Chain state transition diagram for a list of size three, $\{a, b, c\}$, where the sequences in each state of the chain is a permutation of the elements. The adaptive strategy in this example is the MTF rule. 60
2.11 The reduced Markov Chain state transition diagram produced by the MTF rule. The first state represents when list configurations a precedes b, and the second when b precedes a. 63
2.12 The Markov Chain state transition diagram produced by the TR rule. 64
2.13 A diagrammatic description of the Move-To-Front (MTF) rule. 66
2.14 A diagrammatic representation of the Transposition Rule (TR). . . . 67

3.1 Iacono's data structure expressed as a sequence of doubly-exponential queues and the corresponding balanced search trees. The queues arranged in sequence constitute the original list. 83
3.2 A diagrammatic description of the MTF-MTF Hierarchical scheme . . 86
3.3 A diagrammatic description of the MTF-TR Hierarchical scheme . . . 87
3.4 A diagrammatic description of the TR-MTF Hierarchical scheme . . . 88
3.5 A diagrammatic description of the TR-TR Hierarchical scheme 89

4.1 The asymptotic cost ratio of the MTF-MTF-EOMA:MTF in MSE . . 107
4.2 The asymptotic cost ratio of the TR-MTF-EOMA:MTF in MSE . . . 108
4.3 The asymptotic cost ratio of the MTF-TR-EOMA:MTF in MSE . . . 108
4.4 The asymptotic cost ratio of the TR-TR-EOMA:MTF in MSE 109
4.5 Changes in the asymptotic cost of the stand-alone and hierarchical schemes with EOMA in the MSE 110
4.6 Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes augmented with the EOMA in a MSE 111
4.7 The asymptotic cost ratio of the MTF-MTF-EOMA:MTF in PSE . . 114
4.8 The asymptotic cost ratio of the TR-MTF-EOMA:MTF in PSE . . . 115
4.9 The asymptotic cost ratio of the MTF-TR-EOMA:MTF in PSE . . . 116
4.10 The asymptotic cost ratio of the TR-TR-EOMA:MTF in PSE 116
4.11 Changes in the asymptotic cost of the stand-alone and hierarchical schemes with EOMA in the PSE 117
4.12 Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with EOMA in the PSE 118
4.13 Asymptotic cost for Periodic variations of MTF-MTF-EOMA in the Zipf distribution 119
4.14 Asymptotic cost for Periodic variations of TR-MTF-EOMA in the Zipf distribution 119
4.15 Asymptotic cost for Periodic variations of MTF-TR-EOMA in the Zipf distribution 120

4.16 Asymptotic cost for Periodic variations of TR-TR-EOMA in the Zipf distribution 120

5.1 The asymptotic cost ratio of the MTF-MTF-PEOMA:MTF in MSE . 129
5.2 The asymptotic cost ratio of the TR-MTF-PEOMA:MTF in MSE . . 129
5.3 The asymptotic cost ratio of the MTF-TR-PEOMA:MTF in MSE . . 130
5.4 The asymptotic cost ratio of the TR-TR-PEOMA:MTF in MSE . . . 130
5.5 Changes in the asymptotic cost of the stand-alone and hierarchical schemes with PEOMA in the MSE 131
5.6 Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with PEOMA in the MSE 132
5.7 The asymptotic cost ratio of the MTF-MTF-PEOMA:MTF in PSE . 135
5.8 The asymptotic cost ratio of the TR-MTF-PEOMA:MTF in PSE . . 135
5.9 The asymptotic cost ratio of the MTF-TR-PEOMA:MTF in PSE . . 136
5.10 The asymptotic cost ratio of the TR-TR-PEOMA:MTF in PSE . . . 136
5.11 Changes in the asymptotic cost of the stand-alone and hierarchical schemes with PEOMA in the PSE 137
5.12 Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with PEOMA in the PSE 137
5.13 Asymptotic cost for Periodic variations of MTF-MTF-PEOMA in the Zipf distribution 139
5.14 Asymptotic cost for Periodic variations of TR-MTF-PEOMA in the Zipf distribution 139
5.15 Asymptotic cost for Periodic variations of MTF-TR-PEOMA in the Zipf distribution 140
5.16 Asymptotic cost for Periodic variations of TR-TR-PEOMA in the Zipf distribution 140
5.17 The asymptotic cost ratio of the MTF-MTF-TPEOMA:MTF in MSE 145
5.18 The asymptotic cost ratio of the TR-MTF-TPEOMA:MTF in MSE . 145
5.19 The asymptotic cost ratio of the MTF-TR-TPEOMA:MTF in MSE . 146
5.20 The asymptotic cost ratio of the TR-TR-TPEOMA:MTF in MSE . . 146

5.21 Changes in the asymptotic cost of the stand-alone and hierarchical schemes with TPEOMA in the MSE 147
5.22 Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with TPEOMA in the MSE 147
5.23 The asymptotic cost ratio of the MTF-MTF-TPEOMA:MTF in PSE 150
5.24 The asymptotic cost ratio of the TR-MTF-TPEOMA:MTF in PSE . 151
5.25 The asymptotic cost ratio of the MTF-TR-TPEOMA:MTF in PSE . 151
5.26 The asymptotic cost ratio of the TR-TR-TPEOMA:MTF in PSE . . 152
5.27 Changes in the asymptotic cost of the stand-alone and hierarchical schemes with TPEOMA in the PSE 153
5.28 Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with TPEOMA in the PSE 153
5.29 Asymptotic cost for Periodic variations of MTF-MTF-TPEOMA in the Zipf distribution 154
5.30 Asymptotic cost for Periodic variations of TR-MTF-TPEOMA in the Zipf distribution 154
5.31 Asymptotic cost for Periodic variations of MTF-TR-TPEOMA in the Zipf distribution 155
5.32 Asymptotic cost for Periodic variations of TR-TR-TPEOMA in the Zipf distribution 155

1 Introduction

1.1 Introduction

Data storage and processing are the twin pillars of computing. From time immemorial, preceding the advent of modern computing, the need to store data efficiently, and to likewise carry out computation or processing on that data for the purposes of generating information, have been the hallmarks or central concerns of computation. From the modern computer era, which took off with the "Electronic Numerical Integrator and Computer" (ENIAC) and the "Electronic Discrete Variable Automatic Computer" (EDVAC) in the 1940s, advances in computational science, and the corollary opening of new frontiers in other fields that have benefited from computing, have been primarily due to the progress made in the optimization and expansion of the storage and processing capabilities of computational devices.

Apart from the above, research advances in data storage and processing are critical drivers in pushing the envelope for Artificial Intelligence (AI), which is, currently, spawning the "Age of Intelligence". A major concern in incorporating such "intelligence" capabilities into machines and algorithms, and, indeed, in computation, is the issue of "speed". This includes the speed in accessing data and/or in carrying out the prescribed computations. This tacitly involves the concept of "speed" in *retrieving* data so as to minimize its time complexity or computational cost. The need for enhancing speed is critical for almost all applications and domains, and this is most relevant when real-time or near real-time efficiency is desired.

The goal of this research endeavor is to further 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 result of such a process is the subsequent minimization of the cost associated with query accesses.

The Environments under consideration in this work are what we refer to as Non-stationary Environments (NSE), where the access probabilities of the data elements vary with time. These are akin to many real-world scenarios. These Environments also exhibit a particular dependency property called "locality of reference" where the events are probabilistically dependent on one another. We consider two such Environments in this work, 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 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 in [3] for Singly-Linked Lists (SLLs) on Singly-Linked Lists. The LOL concept consists of an outer-list and a sub-list, whose elements are called the outer and sub-list contexts respectively. In this framework, elements that are more likely to be accessed together are grouped within the same sub-context, while the sub-lists are moved "en masse" towards the head of the list-context by following a re-organization rule.

In order to capture the probabilistic dependence of the elements in the data structure, based on the query accesses from the Environment, this work employs a set of reinforcement learning schemes derived from the theory of Learning Automata (LA). These schemes are, at this juncture in time, the state-of-the-art procedures for solving the object partitioning problem explained in detail in Chapter 2. These reinforcement schemes are variants of the so-called "Object Migration Automaton" (OMA).

1.2 Motivation of the Thesis

To motivate this work, consider the time complexity of retrieving an element from a SLL. However, if we can "adaptively" modify the elements of the list such that the frequently-accessed elements are moved towards the head, we can asymptotically do better than this i.e., reduce the constant for the Linear term. Currently, the "de-facto" rules when referring to self-organization or an adaptive list have been the Move-to-Front (MTF) and the Transposition (TR) rules[1] proposed by McCabe in [36]. The MTF rule operates by moving the accessed element to the head of the list, while the TR rule moves the accessed element one-step towards the list head. It has been shown that the MTF rule is characterized by a quicker convergence rate as it rapidly responds to the queries from the Environment. The TR rule, on the other hand, is more conservative in its update approach, and leads to more stable asymptotic costs.

The pros of the MTF and TR rules are combined in designing the hierarchical

[1]Rules other than the MTF and TR have also been proposed and analyzed in the literature. These are surveyed in Chapter 2.

data "sub"-structures alluded to above. These help to mitigate the rapid response of the MTF rule to queries from the Environment, as well as to improve the convergence speed of the TR rule. This idea of combining the MTF and the TR rules is, indeed, not entirely novel in designing ADSs in NSEs. Other researchers have also merged both these rules for precisely the same reasons. This has led to a number of composite MTF and TR strategies such as the Move-ahead-k, MHD(k) [50], where a queried element is moved k positions to the head of the list, the POS(k), where the queried element is moved to the k^{th} position of the list if $j > k$, and otherwise it uses the TR rule so long the element is in positions 2 to k. Another mechanism is the SWITCH(k) rule which is similar to the POS(k) except that it applies the MTF rule when the element is in positions 2 to k. However, these schemes are not as robust as the MTF and the TR rules in NSEs possessing "locality of reference".

It should be mentioned that the MTF rule has been shown to be superior to other re-organization strategies in such NSEs. As a result, the MTF and the TR rules also form the baseline for empirically assessing the quality and efficiency of the hierarchical schemes in this work, namely, those that are augmented by the OMA-family of reinforcement strategies.

1.3 Problem Statement of the Thesis

The MTF and TR re-organization rules discussed above, when used in the context of hierarchical strategies, result in the MTF-preceding-MTF, (MTF-MTF), MTF-preceding-TR, (MTF-TR), TR-preceding-MTF, (TR-MTF), and TR-preceding-TR, (TR-TR) schemes. By way of explanation, in the TR-MTF scheme, the element within a particular sub-context is moved one-step towards the head of the sub-list containing it, while the sub-list itself is moved to the head of the list context. Again, note that the MTF and TR schemes are the "primitives" employed in this hierarchical formulation to take advantage of the fast convergence properties of the MTF rule and the more accurate asymptotic ordering of the TR rule.

The hierarchical schemes presented, however, suffer from a major drawback when used "as-is" in NSEs. Recall that one of the key advantages of the hierarchical

formulation is the "en masse" promotion of list elements within the same sub-list, where the elements in the latter are assumed to have a probabilistic dependence. However, if such a dependence is not "learned", it is far from the case that the elements within the sub-lists are probabilistically dependent. Consequently, in the absence of such a learning mechanism, the hierarchical schemes end-up having an inferior performance to the MTF and TR rules in NSEs.

The pioneering work of Amer and Oommen in [3] utilized the OMA algorithm to capture the probabilistic dependence of the queries coming from the Environment. The introduction of the OMA reinforcement scheme mitigated the static ordering of the sub-lists so that the elements can move freely from one sub-list partition to another as the OMA learns the optimal sub-list grouping. The addition of the OMA to the primitive hierarchical schemes, resulted in the MTF-MTF-OMA, MTF-TR-OMA, TR-MTF-OMA and the TR-TR-OMA, where the third component in the triple is the reinforcement scheme.

Unfortunately, the OMA algorithm used in the hierarchical scheme implementation suffers from a deadlock[2] scenario 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 [21]. 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 certain about the identity of the elements that should constitute a sublist. This work augments the hierarchical schemes with the EOMA reinforcement algorithm.

More recent research has concentrated on further improving the LA's convergence by means of the *Pursuit* Enhanced Object Migration Automaton (PEOMA) and the *Transitivity* Pursuit Enhanced Object Migration Automaton (TPEOMA) reinforcement algorithms respectively. The PEOMA incorporates the Pursuit concept to filter

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

divergent queries from the Environment while the TPEOMA takes advantage of the Transitivity phenomenon based on the statistical distribution of the queried elements to infer good query pairs from non-accessed elements in the transitivity relation. Our research demonstrates how we can design hierarchical SLLs using the PEOMA and the TPEOMA respectively.

1.4 Objectives of the Thesis

From the problem statement discussed above, the key objectives of this thesis are to:

- Design an hierarchical SLL data "sub"-structure using the EOMA reinforcement scheme. This design would yield the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and the TR-TR-EOMA schemes.
- Design analogous hierarchical PEOMA-augmented SLL-based LOLs which would result in the MTF-MTF-PEOMA, MTF-TR-PEOMA, TR-MTF-PEOMA and the TR-TR-PEOMA schemes.
- Design hierarchical TPEOMA-augmented SLL-based LOLs. These would yield the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and the TR-TR-TPEOMA schemes.
- Evaluate the performances of the new schemes in MSEs and PSEs under five different types of query distributions, namely, the Zipf, Eighty-Twenty, Lokta, Exponential and Linear distributions
- Analyze the behaviours of all the newly designed LOLs under varying sublist divisions, as well as under variations of the dependent-factor, α, and the period term, T, which are hyper-parameters of MSEs and PSEs respectively.
- Assess the asymptotic and amortized costs of all the LOLs under the above-mentioned constraints, as well as compare their cost ratios and rates of convergence to the MTF and TR schemes in such Environments.

1.5 Contributions of the Thesis

We can briefly summarize the conclusive results[3] of this research as follows:

- The EOMA-augmented hierarchical schemes performed better than the original OMA-augmented schemes that used the novel idea of a hierarchical LOL approach;
- The EOMA-augmented hierarchical schemes were superior to the MTF and TR rules when the outer-list context was the MTF;
- The PEOMA-augmented hierarchical schemes were as order of magnitude superior to the EOMA-augmented schemes, as well as to the MTF and TR stand-alone rules in MSEs;
- The PEOMA-augmented hierarchical schemes were also superior to the EOMA-augmented schemes, and also superior to the stand-alone MTF and TR schemes in PSEs even when the outer-list context was the TR;
- The TPEOMA-augmented hierarchical schemes also achieved superior performances to the EOMA-augmented schemes, and to the MTF and TR stand-alone rules in MSEs;
- The TPEOMA-augmented hierarchical schemes were shown to be unsuitable for PSEs;
- The "Periodic" and "UnPeriodic" versions of the EOMA and PEOMA-augmented hierarchical schemes yielded a superior performance in PSEs to those without such additions.

In conclusion, the key contribution of this thesis is the enhancement of the hierarchical LOL approach for learning "locality of reference" in NSEs, with the EOMA, PEOMA and TPEOMA reinforcement paradigms. The resultant enhanced hierarchical schemes designed by this research are currently the state-of-the-art methods for adaptive singly-linked lists operating in NSEs.

[3]All of these results which constitute the novel contributions of the thesis, are exclusively due to the work of my supervisor and myself.

1.6 Hardware/Software Used for the Simulations

In the interest of completeness, we provide the details of the platform used in carrying-out the simulations in this work:

- The processor specification for the hardware used for the simulations was a 2 GHz Intel Core i7, with a 8 GB 1600 MHz DDR3 memory;
- The Operating System software was a macOS Sierra;
- The simulations were coded with the Python 3.6 programming language;
- At this juncture, we want to point out that there are two principal methods to evaluate the results for a simulation. One is to compute the means and standard deviations of the results within a margin of error, and the other is to compute the ensemble mean of the results over a large number of experiments. The latter is used in this research.

1.7 Organization of the Thesis

Chapter 2 will survey the theory of LA, covering fixed and variable structure stochastic automata, and the estimator and pursuit concepts. The field of LA forms the framework for the OMA used in learning the true partition of the objects into groups. Also, the estimator and pursuit concepts are the backbone of the PEOMA and TPEOMA algorithms. This chapter will also survey other object partitioning schemes and the fundamental adaptive list organizing schemes in the literature, and their behaviour in NSEs.

Chapter 3 will examine the motivation for designing the hierarchical scheme as a data "sub"-structure. This chapter will introduce the MTF-MTF, MTF-TR, TR-MTF and TR-TR hierarchical schemes, and will explain the drawback of statistically capturing the dependence within sub-lists. This will constitute the justification for introducing the OMA-based reinforcement schemes in the next chapter for dynamic dependence capturing.

Chapter 4 will augment the hierarchical schemes with the EOMA and explain the rationale for the OMA-based enhancement in mitigating the deadlock scenario to ensure a stable convergence to the optimal partitioning of elements. It also covers the modification of the criteria for convergence to include elements that are within a few of the most internal states.

Chapter 5 will augment the hierarchical schemes with the PEOMA and the TPEOMA reinforcement algorithms that incorporate the Estimators and Pursuit concepts to further enhance/improve the dependence-capturing mechanism of the system.

In both Chapters 4 and 5, we will implement the Periodic variants for the LOLs. These variants will consist of using the "Periodic" and "UnPeriodic" additions to the hierarchical schemes, where the LOL data-structure will either be provided with or not provided with the knowledge of the period from the Environment respectively, as the query space changes.

Finally, Appendices A, B, C and D contain the more detailed and comprehensive results of the simulations in Chapters 4 and 5. The results of the original OMA-augmented [3] hierarchical SLLs is presented in Appendix A, included in the interest of completeness. Appendices B, C and D contain the results for the EOMA-augmented, PEOMA-augmented, and TPEOMA-augmented hierarchical SLLs respectively.

2

Background Survey

2.1 Introduction

This chapter surveys[1] the areas that we will be working with, namely the fields of Learning Automata (LA) and Adaptive Data Structures (ADSs). To motivate this research on ADSs, we will survey the field of LA to some depth. The field of LA centers on developing computationally adaptive learning schemes when the environment responds randomly. To explain this, let us assume the existence of an environment (it can be an abstract environment), that is stochastic. This type of environment is akin to many real-world settings. We now place a learning agent that interacts with this environment, which is an automaton or a finite state machine.

In this setting, the automaton is tasked with making decisions by interacting with

[1]The chapter, in and of itself, is comprehensive, and in thus a little long. Our aim is that it will serve as a fairly exhaustive exegesis of the respective fields, rendering the thesis to be a stand-alone document.

this environment. The question that arises is that of determining how the automaton can learn the best action offered by the random environment? The objective of Learning Automata is to design an algorithm or a learning scheme that is expedient (i.e., better than doing nothing) at the very least. Indeed, more than being expedient, many learning schemes from this field have been proven to be ϵ-optimal, i.e., as close as being optimal as desirable.

In this chapter, we will review the LA task specification model, the families of Fixed-Structure and Variable-Structure Stochastic Automata learning schemes, as well as the Estimator and Pursuit schemes for both stationary and non-stationary environments.

After this, we will examine the Object Partitioning Problem (OPP) approached from an LA perspective. The OPP is a well-known NP-hard problem, whose solution complexity is exacerbated when attempting to partition objects in a stochastic environment. Here, we survey various solutions including the "Transitive Pursuit Enhanced Object Migration Automaton" (TPEOMA) which incorporates the transitive and pursuit learning phenomena into the enhanced Object Migration Automaton (OMA) to mitigate the effects of partition-performance degradation due to uncertainties in queries emanating from a stochastic environment.

Finally, we will survey the multiple categorizations that exist in research on self-organizing or ADSs from the literature. ADSs can be surveyed based on their underlying pseudo-physical or mathematical formulations. The data architecture can be in the form of lists or other linear formations, or as trees for two-dimensional structures. Another critical factor for ADS classification involves the memory requirements needed for the self-organization. This takes into consideration the memory demands of the data structure in the adaptation process. Some ADSs require zero or constant $O(1)$ additional memory, while others may require $O(n)$, i.e., linear extra memory.

What is, perhaps, most pertinent to this research, is the categorization based on the statistical relationship of the query accesses. The queries generated are said to be independently and identically distributed (i.i.d) if we assume that every element in the sequence is independent of all the preceding elements. Otherwise, the queries are dependent - analogous to a Markovian sequence [12] where an element is statistically

dependent on the previous element in the sequence. Environments that have this dependence property, which we shall refer to as "locality of reference", are particularly crucial to the ADS formulation undertaken in this research.

Cataloging the schemes with respect to the statistical relationship of the query access patterns for ADSs is further discussed from the perspective of stationarity and non-stationarity of the environment itself. We model stationary environments by assuming that the states of the environment do not change with time, i.e., that the states have a deterministic, probabilistic distribution. As opposed to this, in Non-stationary Environments (NSEs), the states of the environment alter with time.

The Markovian Switching Environment (MSE) and the Periodic Switching Environment (PSE) are two critical models for Environments exhibiting non-stationarity with probabilistically-dependent components. These models are critical to formulating our hierarchical ADSs.

2.2 Learning Automata

Learning automata (LA) arose in the Soviet Union in the 1960s by Tsetlin as a computational adaptive scheme for learning. He proposed a model for developing a strategy or policy for choosing an optimal action from a finite-hypothesis space of legal actions in stochastic (non-stationary) environments. LA share multiple interfaces in broad concepts with the fields of reinforcement learning, multi-agent systems, and control theory.

The LA task can be modeled by means of a feedback loop between the Environment and the automaton (i.e., the decision maker) as in Figure 2.1. The automaton interacts with the Environment by choosing from a set of actions based on the feedback it receives from the Environment. The concept of the "feedback signal" as a reward or penalty is derived from learning theories in conditioning and associative learning, with its computational approaches well documented in [66]. The LA model is set up as an adaptive process [6] in which little or no information is known *a priori* about the Environment. The goal of the learner (the LA) is then to learn the optimal action that maximizes a utility function or improves a performance index.

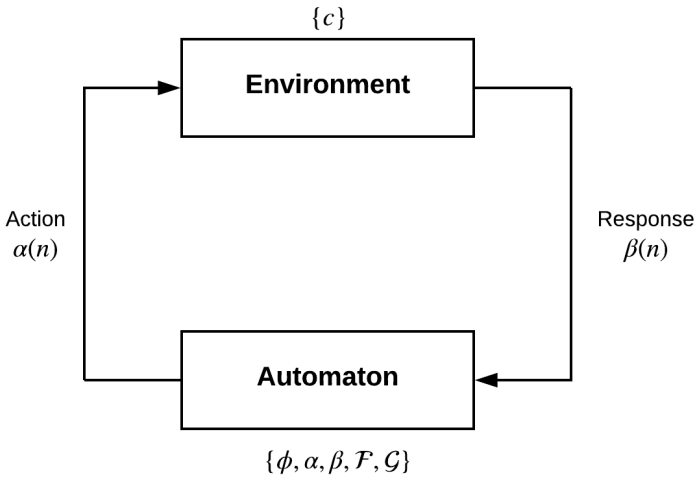


Figure 2.1: The Learning Automata Model

The two major constituents of the learning model are the Environment and the automaton.

The Environment

The notion of an Environment in the LA design can be perceived as an abstract stochastic medium with which the LA interacts. Formally, the Environment is represented as a triple:

$$\mathbb{E} = \{\underline{\alpha}, \underline{c}, \underline{\beta}\}$$

where,

- $\underline{\alpha}$ represents a set of r inputs and is the set of actions that the LA can choose from, i.e., $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_r\}$.
- $\underline{\beta}$ represents the "feedback signal" or response from the Environment. The response is a binary set consisting of a reward or a penalty, $\underline{\beta} = \{0, 1\}$. This instance of a binary response set is also known as the P-model. The response set can further be generalized to allow a finite output set with more than two elements in the $[0, 1]$ interval, or it can be a continuous random variable also

in the $[0, 1]$ interval. The former is known as the Q-model, while the latter is referred to as the S-model.

- $\underline{c}$ represents the set of penalty probabilities that correspond to each input action into the Environment from the automaton. More specifically,

$$Pr(\beta(n) = 1 \mid \alpha(n) = \alpha_i) = c_i \quad (i = 1, 2, ..., r).$$

A stationary Environment is characterized by penalty probabilities that are constant. For NSEs, the penalty probabilities vary with time.

The Automaton

The Automaton is the key mechanism for choosing the set of actions or policies based on the responses from the Environment so as to maximize some value or utility function. In LA design, the automaton is described as a quintuple:

$$\mathbb{A} = \{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, \mathcal{F}(.,.), \mathcal{G}(.,.)\}$$

where,

- $\underline{\Phi}$ represents a finite set of the internal states of the automaton. At any time instant n, the LA's state is denoted as $\phi(n)$. Formally, $\underline{\Phi} = \{\phi_1, \phi_2, ..., \phi_s\}$.
- $\underline{\alpha}$ represents a finite set of actions that the LA can choose from. At any time instant n, the action chosen is expressed as $\alpha(n)$. Formally, $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_r\}$.
- $\underline{\beta}$ represents the input to the automaton or the feedback from the Environment. This quantity is identical to the set $\underline{\beta}$ defined for the Environment.
- $\mathcal{F}(.,.)$ represents a mapping (that is either deterministic or stochastic) from the current state $\phi(n)$ to the next state, i.e., $\phi(n+1) = \mathcal{F}[\phi(n), \beta(n)]$. Formally, $\mathcal{F} : \underline{\Phi} \times \underline{\beta} \rightarrow \underline{\Phi}$.
- $\mathcal{G}(.,.)$ defines the output mapping (that is either deterministic or stochastic) of the automaton at an instant n. This also represents the way by which the action is chosen, based on the state of the Environment at n. Thus, $\alpha(n) = \mathcal{G}[\phi(n)]$, and so, $\mathcal{G} : \underline{\Phi} \rightarrow \underline{\alpha}$.

The performance of the automaton is summarized with the following accepted criteria [37]. As per [37], the quantity $M(n)$ defines the average penalty for a given action probability vector, $P(n) = [p_1(n), \ldots, p_R(n)]$ as:

$$\begin{aligned} M(n) &= \mathbb{E}[\beta(n)|P(n)] = Pr[\beta(n) = 1|P(n)] \\ &= \sum_{i=1}^{r} Pr[\beta(n) = 1|\alpha(n) = \alpha_i] Pr[\alpha(n) = \alpha_i] \\ &= \sum_{i=1}^{r} c_i p_i(n). \end{aligned}$$

The quality of learning can be evaluated by comparing the performance with a pure-chance automaton, M_0. A pure-chance automaton is a random choice machine and occurs when there is no basis to differentiate between the actions $\alpha_i (i = 1, 2, ..., r)$. Since each action is selected with an equal probability, M_0 has the form:

$$M_0 = \frac{1}{r} \sum_{i=1}^{r} c_i.$$

An LA that performs reasonably well must asymptotically at least do better than pure chance. Hence, we compare $\mathbb{E}[M(n)]$ with M_0, which we do by observing:

$$\begin{aligned} \mathbb{E}[M(n)] &= \mathbb{E}\{\mathbb{E}[\beta(n)|P(n)]\} \\ &= \mathbb{E}[\beta(n)] \end{aligned}$$

Hence,

1. The LA is said to be *Expedient* if: $\lim_{n\to\infty} \mathbb{E}[M(n)] < M_0$.
2. The LA is *optimal* if: $\lim_{n\to\infty} \mathbb{E}[M(n)] = c_\ell$, where, $c_\ell = \min_i c_i$.
3. The LA is ϵ-*optimal* if: $\lim_{n\to\infty} \mathbb{E}[M(n)] < c_\ell + \epsilon$, where ϵ is an arbitrary value greater than zero.
4. Finally, a LA is *absolutely expedient* if $\mathbb{E}[M(n+1)|P(n)] < M(n)$, and thus $\mathbb{E}[M(n)]$ is continually decreasing with n in all stationary stochastic Environments.

The following sub-sections will briefly review the families of *Fixed-Structure Stochastic Automata* (FSSA) - when the intra-state transitions are constant irrespective of the input or states, and the families of *Variable-Structure Stochastic Automaton* (VSSA) - when the transition probabilities change, possibly at every time instant. We will also survey estimator and pursuit schemes, as well as some models of non-stationarity.

2.2.1 Fixed Structure Stochastic Automaton

The kernel of the analysis of FSSA is based on the notion that fixed-structure schemes can be characterized via homogeneous Markov chains that are ergodic and hence have a final solution that is independent of the LA's starting states.

The first of such schemes, the two-state automaton[2], $L_{2,2}$, was proposed by Tsetlin in 1961 [61]. As it is a special case of the next machine studied, the $L_{2N,2}$, we will not explain it any further.

Tsetlin's FSSA with Memory

The $L_{2N,2}$ automaton (also proposed by Tsetlin) has $2N$ states, $\{\phi_1, \phi_2, ..., \phi_{2N}\}$ and 2-actions, α_1 and α_2. It takes into consideration the previous behavior of the system in determining the series of actions chosen. In keeping with fixed-structure schemes, it is analyzed using the theory of Markov chains.

Consider the structure of the $L_{2N,2}$ given in Figure 2.2. The states $\{\phi_1, \phi_2, ..., \phi_N\}$ belong to α_1, and the states $\{\phi_{N+1}, \phi_{N+2}, ...\phi_{2N}\}$ belong to α_2. Observe the ingenious, non-trivial ordering of the state in the layout of the Markov chain. The $L_{2N,2}$ automaton functions by advancing farther into the memory of an action when it is rewarded, and moving towards the boundary states on being penalized.

By deriving the equilibrium state probabilities and the asymptotic action probabilities, one can see that the expected asymptotic penalty $M(\infty)$ of the $L_{2N,2}$ machine has the form $M_{L_{2N,2}}$ given below:

[2]This simple learning heuristic, also known as"run and twiddle" has found relevance in many natural and artificial systems [53]. From analysis, since the transition matrices are deterministic, the $L_{2,2}$ automaton is expedient, i.e., $M(\infty) < M_0$, but not ϵ-optimal.

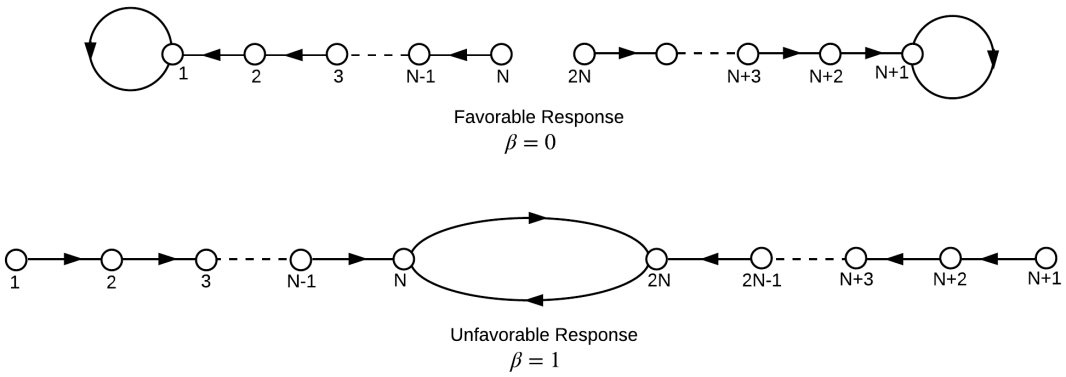


Figure 2.2: The transition map of the $L_{2N,2}$ Automaton

$$M(L_{2N,2}) = \frac{\frac{1}{c_1^{N-1}} \; \frac{c_1^N - d_1^N}{c_1 - d_1} + \frac{1}{c_2^{N-1}} \; \frac{c_2^N - d_2^N}{c_2 - d_2}}{\frac{1}{c_1^N} \; \frac{c_1^N - d_1^N}{c_1 - d_1} + \frac{1}{c_2^N} \; \frac{c_2^N - d_2^N}{c_2 - d_2}}.$$

The derivation of the above expression is rather contrived and non-trivial, and the reader is referred to [37] for the details of the analysis. This is a monotonically decreasing function of the memory depth, N, and it converges to the minimum penalty probability as long as the latter is less than 0.5, and so:

$$\lim_{N \to \infty} M(L_{2N,2}) = \min(c_1, c_2)$$

when $\min(c_1, c_2) < 0.5$.

For systems with r-actions ($r > 2$), $M(L_{rN,r})$ leads to a comparable expression which has a limit of $\min\{c_1, c_2, ..., c_r\} = c_\ell$, whenever $c_l < 0.5$, which is thus the necessary condition for ϵ-optimality. This condition embodies a notable limitation, as it does not guarantee an asymptotic "perfect" convergence in all random environments.

ϵ-Optimal FSSA

Optimality, in LA implies that the automaton must somehow figure out how to learn the action with the minimum penalty probability. While this is, in general, a gold standard, it is unattainable.

As opposed to this, ϵ-optimality is an attempt to make the automaton as close to optimal as possible. This is achieved by modifying a parameter setting in the learning scheme. The parameter, in the context of a deterministic scheme, is the memory depth associated with each action.

A FSSA is ϵ-optimal, if for every $\epsilon > 0$, there exists an N_1 such that for $N > N_1$

$$M \leq \min_i\{c_i\} + \epsilon$$

where, N is the memory depth of the automaton and c_i for $0 < i \leq r$ lies in the closed interval $[0, 1]$. The Krinsky [31] and Krylov [32] automata are examples of ϵ-optimal schemes.

The update scheme for Krinsky's automaton is shown in Figure 2.3 below. On penalty, Krinsky's ϵ-optimal scheme is similar to the $L_{2N,2}$ automaton. But on reward, the automaton jumps to the innermost state of the action. Consequently, on reward, an automaton in any of the states $\phi_1, ..., \phi_N$, moves to ϕ_1, else, if it is in any of states $\phi_{N+1}, ..., \phi_{2N}$, it moves to ϕ_{N+1}. The expected penalty for Krinsky's automaton, $M(K_{r1,2N,2})$ is given below, and the reader should refer to [37] for the complete analysis.

$$M(K_{r1,2N,2}) = \frac{c_1 c_2^N + c_2 c_1^N}{c_1^N + c_2^N} = \frac{\frac{1}{c_1^{N-1}} + \frac{1}{c_2^{N-1}}}{\frac{1}{c_1^N} + \frac{1}{c_2^N}}.$$

As the memory depth, N, tends to infinity, $\lim_{n\to\infty} M(K_{r1,2N,2}) = \min(c_1, c_2)$.

Consider now the Krylov LA, whose transition map is shown in Figure 1.4. On reward, Krylov's strategy is similar to the $L_{2N,2}$ automaton. But on penalty, the automaton moves either way in the state-space of the corresponding action with probability 0.5. So, on penalty, an automaton in states ϕ_i, where $\phi_i \neq \{1, N, N + 1, 2N\}$, moves to state ϕ_{i+1} or to state ϕ_{i-1} with probability 0.5. If $\phi_i = \{1, N + 1\}$, the automaton stays in its current state, or move to state ϕ_{i+1} with probability 0.5. Lastly, if $\phi_i = \{N\}$, the automaton moves to state ϕ_{N-1} or to ϕ_{2N} with probability 0.5, and if $\phi_i = \{2N\}$, the automaton moves to state ϕ_{2N-1} or ϕ_N with the same probability. The expected asymptotic penalty for Krylov's automaton, $M(K_{r2,2N,2})$, is presented below, and the analysis for this machine is also found in [37].

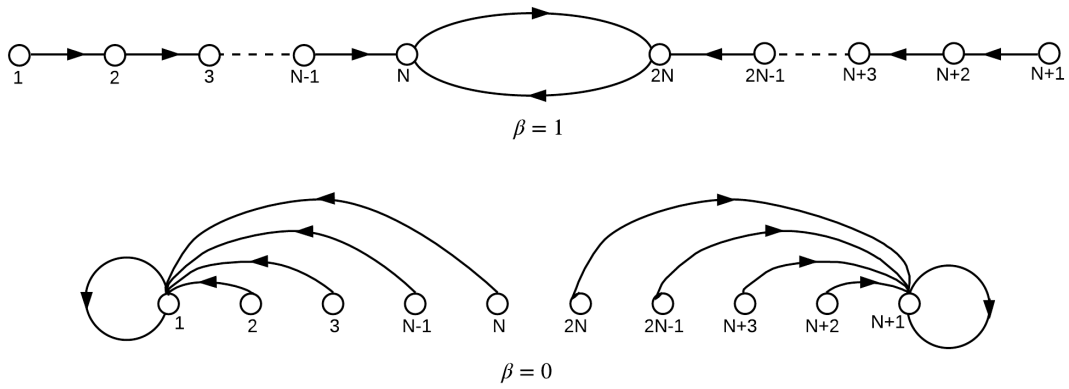


Figure 2.3: The transition map of Krinsky's Automaton

$$M(K_{r2,2N,2}) = \frac{\frac{1}{\lambda_1^{N-1}} \ \frac{\lambda_1^N - 1}{\lambda_1 - 1} + \frac{1}{\lambda_2^{N-1}} \ \frac{\lambda_2^N - 1}{\lambda_2 - 1}}{\frac{1}{c_1^N} \ \frac{1}{\lambda_1^{N-1}} \ \frac{\lambda_1^N - 1}{\lambda_1 - 1} + \frac{1}{c_2^N} \ \frac{1}{\lambda_2^{N-1}} \ \frac{\lambda_2^N - 1}{\lambda_2 - 1}},$$

where, $\lambda_i = \frac{c_i}{1+d_i}$. Again, as the memory depth, N, tends to infinity, $\lim_{n\to\infty} M(K_{r2,2N,2}) = \min(c_1, c_2)$.

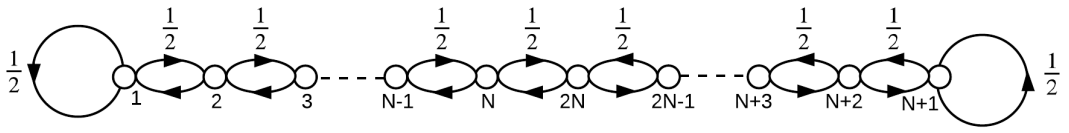


Figure 2.4: The transition map of Krylov's Automaton for receiving a penalty

The FSSA schemes presented above can easily be extended to r-actions by appropriately modifying their transitions on receiving a penalty or a reward. The key distinction is in how the automaton moves from one action to another in the boundary states. The move to another action from a boundary state can follow a fixed sequence, i.e., $\phi_N \rightarrow \phi_{2N} \quad \phi_{2N} \rightarrow \phi_{3N} \quad \phi_{3N} \rightarrow \phi_{(R-1)N} \quad \phi_{(R-1)N} \rightarrow \phi_{RN} \quad \phi_{RN} \rightarrow \phi_N$. Alternatively, the automaton can move to the boundary of any of the other $r-1$ actions with a certain probability (referred to as the "beta" automaton by Tsetlin).

FSSA form a sophisticated class of adaptive learning schemes and have been employed in solving a variety of NP-hard optimization problems [14, 15]. Of particular interest to the research that we have undertaken is its use to address the object partitioning problem by formulating the Object Migration Automaton (OMA) [47, 46]. OMA-based LA will be described, in detail, in the coming sub-sections.

2.2.2 Variable Structure Stochastic Automaton

VSSA incorporate more plasticity to the learning model by making both or either of the state transition and/or the action probabilities stochastic by using a reinforcement scheme [64]. However, working with action probabilities render the analysis more mathematically intricate [16, 17, 18, 19, 20], and the theory of (small step) Markov processes is central to the analysis of VSSA.

For ease of mathematical expressiveness, one assumes that the states and actions to mean the same entity, and hence, every state represents a distinct action. In that case, using the notation for actions and states, $r = s < \infty$ and the mapping G, specified for LA, is the identity mapping. The automaton is now represented as a triple $\{\underline{\alpha}, \underline{\beta}, \mathcal{A}\}$, where $\mathcal{A}$ is an update or reinforcement scheme.

VSSA update their action probabilities based on their interplay with the Environment and the response that they receive as inputs. The updating principle can be linear or non-linear.

Reinforcement Schemes

The general framework for a reinforcement scheme is formalized by [37] for updating action and transition probabilities. The formal description for updating the action probability vector is given as:

$$P(n+1) = \mathcal{T}[P(n), \alpha(n), \beta(n)], \tag{2.1}$$

where $\mathcal{T}$ is the mapping alluded to above. The action probability at instant $n+1$ is updated based on the vector value, $P(n)$, the action chosen, $\alpha(n)$, and the input received, $\beta(n)$, at the instant n.

As opposed to this, one observes that while the formal description for updating transition probabilities is given as:

$$f_{ij}^{\beta}(n+1) = \mathcal{T}'[f_{ij}^{\beta}(n), \phi(n), \phi(n+1), \beta(n)], \tag{2.2}$$

where $\mathcal{T}'$ is the mapping, and the transition probability at instant $n+1$ for input β representing a transition from state ϕ_i to ϕ_j is updated based on the state of the automaton at $n+1$, (i.e., $\phi_{(n+1)}$), together with its value, the state of the automaton, $\phi(n)$, and the input, $\beta(n)$ at the previous instant, n. Equation (2.1) is the expression that is utilized the literature rather than Equation (2.2).

Reinforcement schemes are categorized based on:

1. The asymptotic behavior of an automaton that uses the scheme, i.e., whether it is expedient, ϵ-optimal, or Absolutely Expedient.

2. The nature of the mapping $\mathcal{T}$ or $\mathcal{T}'$, e.g., linear, non-linear or hybrid. If $P(n+1)$ is a linear function of $P(n)$, the scheme is linear; otherwise, it is non-linear. A hybrid-scheme combines them to take advantage of both these properties. The scheme thus operates using the values of $P(n)$ in one way or another.

3. The properties of the Markov process describing the automaton, i.e., whether it is ergodic or absorbing.

Linear/ Non-linear Schemes

Linear schemes possess the distinct property of being more analytically tractable. Three prominent linear schemes with properties that exemplify the behaviors observed in LA are the Linear Reward-Penalty scheme (L_{R-P}), the Linear Inaction-Penalty scheme (L_{I-P}) and the Linear Reward-Inaction scheme (L_{R-I}) [54, 39, 38, 17, 18, 40, 33, 9].

In VSSA, the transition probabilities are updated using explicit updating formulae. The linear schemes update their action probabilities based on the linear update rule. The expressions below formalize the generalized update of action probabilities in both linear and non-linear VSSA [37].

If

$$\begin{aligned} \alpha(n) &= \alpha_i \qquad (i = 1, 2, ..., r) \\ p_j(n+1) &= p_j(n) - g_j[p(n)] \quad \text{when } \beta(n) = 0, \\ p_j(n+1) &= p_j(n) + h_j[p(n)] \quad \text{when } \beta(n) = 1. \end{aligned}$$

for all $j \neq i$.

For preserving the consistency of the probability measure that requires that $\sum_{j=1}^{r} p_j(n) = 1$, we have:

$$\begin{aligned} p_i(n+1) &= p_i(n) + \sum_{j=1, j\neq i}^{r} g_j(p(n)) \quad \text{when } \beta(n) = 0, \\ p_i(n+1) &= p_i(n) - \sum_{j=1, j\neq i}^{r} h_j(p(n)) \quad \text{when } \beta(n) = 1. \end{aligned}$$

For the functions g_j and h_j, when $j = (1, 2, ..., r)$, the test of validity are conditioned after the following assumptions:

$$\begin{aligned} &\textit{Assumption 1:} = g_j \text{ and } h_j \text{ are continuous functions} \\ &\textit{Assumption 2:} = g_j \text{ and } h_j \text{ are non-negative functions} \\ &\textit{Assumption 3:} = 0 < g_j(p) < p_j \\ &\qquad\qquad = \sum_{j=1, j\neq i}^{r} [\, p_j + h_j(p(n)) \,] < 1, \end{aligned}$$

for all $i = 1, 2, ..., r$, and all p whose elements are all in the open interval $(0, 1)$.

If g_j and h_j are linear functions, then

$$\begin{aligned} g_j[p(n)] &= a p_i(n) \quad 0 < a < 1, \\ h_j[p(n)] &= \frac{b}{r-1} - b p_j(n) \quad 0 < b < 1. \end{aligned}$$

So, if $\alpha(n) = \alpha_j$,

$$\left.\begin{array}{c} p_i(n+1) = p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) = (1-a)p_j(n),\ j \neq i \end{array}\right\} \quad \beta(n) = 0,$$

$$\left.\begin{array}{c} p_i(n+1) = (1-b)p_i(n) \\ p_j(n+1) = \frac{b}{r-1} + (1-b)p_j(n),\ j \neq i \end{array}\right\} \quad \beta(n) = 1.$$

Given the above expression from [37], the L_{R-P} scheme is at best expedient when $a = b$. But when $b < a$, the L_{R-P} becomes ϵ-optimal as $a \to 0$ and $b << a$.

The L_{R-I} scheme is represented by an absorbing probability Markov chain and updates the action probability on reward but does nothing on penalty. It has been proven to be ϵ-optimal. Also, when $b = 0$, the L_{R-P} scheme degenerates to the L_{R-I} scheme. The L_{I-P} scheme, on the other hand, is represented by an ergodic probability Markov chain and updates the action probability on penalty but does nothing on reward. The L_{I-P} is proven to be expedient, but never ϵ-optimal. However, when the Markov chains are discretized and made artificially absorbing, the L_{R-P} and L_{I-P} schemes become ϵ-optimal.

Two prominent non-linear VSSA are the quadratic scheme [64] and the non-linear two action method [65]. A challenge with non-linear schemes is the difficulty in carrying out analytical research on their convergence capabilities.

2.2.3 Estimator and Pursuit Schemes

Estimator and pursuit schemes employ Maximum Likelihood (ML) or Bayesian estimators to make use of estimates of the reward probabilities to update the actions. MLEs do this by keeping an accumulator that records the number of times a chosen action receives a corresponding reward.

Additionally, for every iteration, the estimated reward vector probability is employed in order to update the action probability vector rather than merely use the feedback from the environment. This strategy minimizes the likelihood of selecting actions with lower reward estimates and increases the probability of choosing actions with higher reward estimates.

The pursuit technique, as a simple case, extends the estimator strategy to further increase the likelihood of selecting actions with higher reward estimates by "pursuing" the currently known "best" action [60]. The pursuit schemes are proven to be ϵ-optimal. Further, the discretized versions of the pursuit technique are not only proven to be ϵ-optimal; they also perform better than their continuous variants.

2.2.4 Non-stationary Environments

Non-Stationary Environments (NSEs) deal primarily with learning in settings that change with time. Thus, a NSE has its penalty probabilities $\{c_i(n)\}$, in which $c_i(n)$ change with time. The changes in the environment are reflected by the optimality of the ordering of the actions changing, and the performance measure $\mathbb{E}[M(n)]$ also varying. Learning schemes with fixed policies may become non-expedient over time and being inadequate for such environments. Of course, the overall goal is to have schemes which possess enough flexibility so as to be able 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).

Concerning the concept of the LA's performance in NSEs, the LA attempts to minimize the overall cost over time as:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{i=1}^{r} \mathbb{E}[\beta(n)].$$

In the MSE, the automaton is modeled to operate in a finite number of environments $\{E_1.E_2, ...E_d\}$, which are themselves considered as states of a Markov chain. Let $p_i(n)$ be the probability of selecting the i^{th} action, α_i, at an instant n, for an automaton with r actions. Then, if $q_j(n)$ is the probability that the MSE is in state E_j, and c_i^j is the penalty probability that is associated with the action a_i in Environment E_j, the average penalty $M(n)$ is seen to be:

$$M(n) = \frac{1}{T} \sum_{i=1}^{r} p_i(n)[\ \sum_{j=1}^{d} q_i(n) c_i^j\ ],$$

and the LA is expedient if

$$\lim_{T\to\infty} \mathbb{E}[M(n)] < \frac{1}{T}\sum_{i=1}^{r}\sum_{j=1}^{d} q_i^* c_i^j,$$

where $\{q_j^*\}$ are the stationary probabilities of the MSE.

In a PSE with d stationary Environments $\{E_1.E_2, ...E_d\}$, the time period T where $T \geq d$ can be known *a apriori* or unknown. If T is known, then T automata are deployed to the T Environments when each one operates once in T instants. A switching device connects the automata in sequence to the Environment at every instant. In this setup, every automaton is working in a stationary Environment and attempts to converge to the "best" action within that setting [37].

When T is unknown, the analysis becomes far more complicated. Two solutions which require a two-level organization of LA have been proposed in the literature, one for use with deterministic FSSA, and the other with VSSA.

In the deterministic case [63], the Environment requires a two-level automata hierarchy, where at the first level $\mathbb{A} = \{\mathbb{A}_1, \mathbb{A}_2, ..., \mathbb{A}_{T_{MAX}}\}$, and at the second level, $\mathbb{B} = \{\mathbb{B}_1, \mathbb{B}_2, ..., \mathbb{B}_{T_{MAX}}\}$. Here, T_{MAX} is the maximum period of the Environment with respect to the number of discrete intervals, and hence there exists a one-to-one relationship between the LA in both levels. This mode of operation is shown in Figure 2.5.

Depending on the choice of the action chosen by the second-level automaton $\mathbb{B}$, the corresponding first-level automaton $\mathbb{A}$ is connected to the Environment in question, sequentially. After all the second-level automata that choose a particular r action are identified, they are penalized with a probability that equals the expected penalty from the corresponding first-level automata. This is exemplified in Figure 2.5.

The VSSA for NSEs consist of a single automaton in the first level with T_{MAX} actions that correspond to the value of the periodicity, and it is responsible for deciding the value of the unknown period. The second level has T_{MAX} automata $\mathbb{B} = \{\mathbb{B}_1, \mathbb{B}_2, ..., \mathbb{B}_{T_{MAX}}\}$ and is designed in a manner that if the period $T(n)$ is selected in the first level, it is only the corresponding automata in the second level that operates in a sequential manner, as shown in Figure 2.6

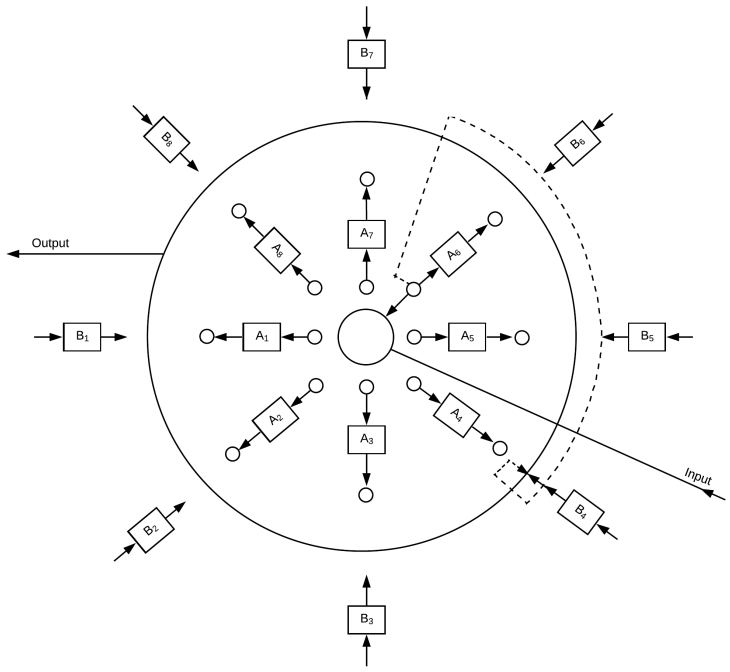


Figure 2.5: The model for a PSE in the case of FSSA.

The average penalty, $\bar{M}(n)$ in a periodic Environment is given as:

$$\bar{M(n)} = \frac{1}{T}\sum_{i=1}^{r}\sum_{t=n}^{n+T-1} c_i(t)p_i(t)$$

2.3 Strategies for Solving the Object Partitioning Problem

This section reviews a very important component of this research, namely the partitioning solution to the Object Partitioning Problem (OPP) useful for processing dependent queries. We shall refer to this phenomenon as the "locality of reference". The following sub-sections review the Object OPP, and the various strategies reported in the literature for achieving this. In particular, we focus our attention on the Object Migration Automaton (OMA), which is the LA approach to resolving the

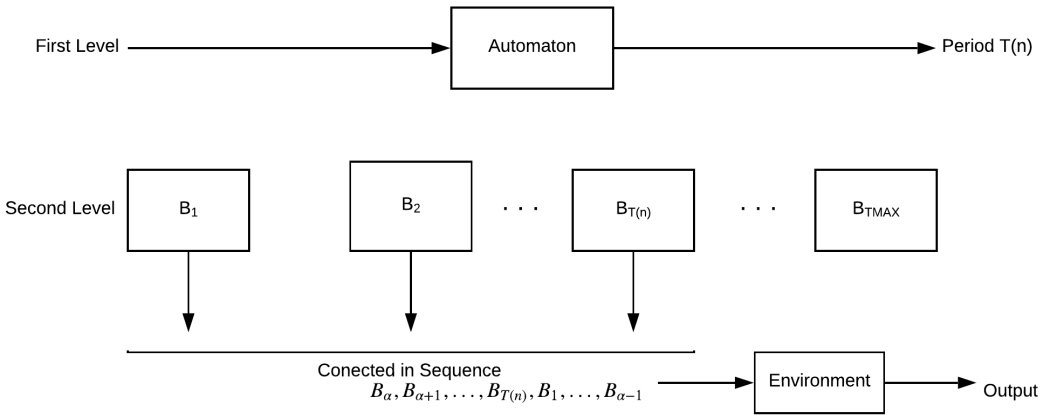


Figure 2.6: The model for a PSE in the case of VSSA.

OPP. Finally, we review the various recent improvements of the OMA, by considering the enhanced versions obtained by incorporating the "pursuit" paradigm and the property of transitivity.

2.3.1 The Case for Partitioning

The need for partitioning is one that naturally emerges in many application domains. Let us suppose that we want to share a dataset across distributed machine clusters, and that we want to know which parts of the dataset should be kept in a particular cluster so as to optimize the operation and/or the performance of the application. The question that arises is to determine how we can design this partitioning mechanism to be automated and intelligent. Of course, the goal is that it captures the dependence between the data accesses and locations so that similar elements are placed together in such a way so as to asymptotically optimize the system's operations. The partitioning problem turns out to be a NP-hard problem and does not have a polynomial time solution [69].

The OPP with respect to the design of an adaptive data structure is to partition a set of W data elements into R disjoint subsets in such a way that elements with that are frequently accessed together are grouped in the same sub-structure. The

probability distribution of the query elements is unknown to the migration automaton.

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 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 [28, 62].

- 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 [48, 49].

- A distributed database system is a conceptual graph-like connection of distinct databases which 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 [35].

- In reputation systems, which takes 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 [67].

- 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 [42, 43].

- 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 [14, 15].

The reader will surely observe that the OPP has close similarities to clustering. Indeed, the OPP can be seen as a clustering algorithm that is based on grouping frequently-accessed elements together.

Partitioning by estimating request statistics, based on the frequency of access counts for a query sequence, can be seen to be a rudimentary solution to the OPP [34]. However, this technique is, obviously, computationally expensive as the number of partitions increases. Further, as we also require the estimates to converge, the number of queries must tend to infinity to guarantee this. Consequently, algorithms employ heuristics to limit the exponential growth of the time taken for the solution.

The partitioning of objects into equally-sized subsets is known as the Equi-Partitioning Problem (EPP). The OMA, that was first proposed in [47], has, since then, emerged to be a very efficient algorithm to resolve the EPP.

In the EPP, the W elements are divided into R groups such that each group has exactly $M = \frac{W}{R}$ objects. The following restrictions characterize the formulation of the EPP:

1. There are equal number of objects in every class.

2. There are two objects in each query. This constraint can easily be relaxed.

The EPP can be formalized as the task of dividing a set $\mathcal{A} = \{A_1, \cdots, A_W\}$ elements, into R groups, specified as $\Omega = \{G_1, \cdots, G_R\}$. Let the true (and optimal) partition of the set into the R groups be designated as Ω^*, where $\Omega^* = \{G_1^*, G_2^*, \cdots G_R^*\}$. Ω^* is unknown to the partitioning algorithm. The objective of the partitioning algorithm is to divide $\mathcal{A}$ into the R groups by means of a partition Ω such that Ω converges to Ω^* as the learning process proceeds. An example of this is shown in Figure 2.7, where the $W = 9$, and the number of classes, $R = 3$. Here, the initial partitioning is given by Ω^0, in the middle block, and the final partition is Ω^+ which, hopefully, becomes Ω^*.

The partitioning is, typically, done by operating on an abstract set of objects $\mathcal{O}$, where $\mathcal{O} = \{O_1, \cdots, O_W\}$, where each abstract O_i maps uniquely to a real world object, A_i. By operating and migrating the abstract objects, $\mathcal{O}$, the real set of

elements, $\mathcal{A}$, would not be manipulated[3]. After a duration of T time instances, the physical objects are updated to harmonize with the partitions of their abstract counterparts.

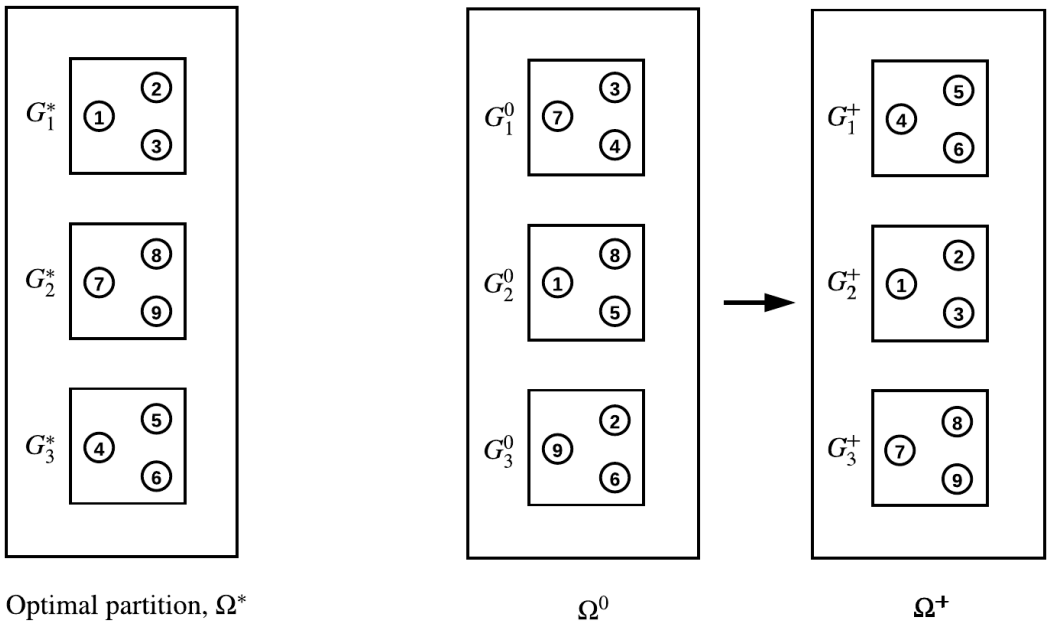


Figure 2.7: The Partitioning Algorithm

We shall now review the existing partitioning strategies for the OPP and the EPP, and we will start with the Basic Adaptive Method (BAM) proposed by Yu, *et al.* [68, 69].

2.3.2 The Basic Adaptive Method (BAM)

The BAM is an improvement on the obvious partitioning solution discussed earlier. It functions by partitioning the abstract objects, $\{O_1, \cdots, O_W\}$ associated with $\mathcal{A}$, into their relevant groups. For many years, the BAM was the state-of-the-art partitioning algorithm. However, the LA-based solutions (i.e., the OMA, and its enhancements)

[3]That being said, our adaptive hierarchical list implementation employs the OMA as a partitioning algorithm applicable for the real-time list re-organization in dependent environments.

that will be discussed presently, are now recorded as the best set of schemes for the EPP.

Algorithm 1 BAM algorithm

Input:

- The abstract objects $\{O_1, ..., O_W\}$,
- The two parameters Δ_1 and Δ_2,
- A stream of queries which every query is conceptually a pair $\langle A_i, A_j \rangle$.

Output:

- A periodic clustering of the objects into R partitions.

```
1: begin
2:     Initialize X_i arbitrarily, where −∞ < X_i < ∞ , for all 1 ≤ i ≤ W.
3:     loop
4:         for a sequence of T queries do do
5:             Read query ⟨A_i, A_j⟩
6:             if X_i < X_j then
7:                 X_i = X_i + Δ_1
8:                 X_j = X_j − Δ_1
9:             else
10:                X_i = X_i − Δ_1
11:                X_j = X_j + Δ_1
12:            end if
13:            Select randomly distinct indices p, q, where 1 ≤ p, q ≤ W
14:            if X_p < X_q then
15:                X_p = X_p − Δ_2
16:                X_q = X_q + Δ_2
17:            else
18:                X_p = X_p + Δ_2
19:                X_q = X_q − Δ_2
20:            end if
21:        end for
22:        Print out partitions based on proximity of {X_i}
23:    end loop
24: end
```

Source: Excerpt from [55]

In the BAM, the queried elements are processed as a pair. Every abstract object, O_i, for a particular query element is designated by a real number X_i. So when record

A_i and A_j are queried, the real numbers X_i and X_j are moved towards their centroid, i.e., $\frac{X_i+X_j}{2}$. This move is controlled by a constant Δ_1. But to prevent all the objects from coming into the same group, two objects are randomly selected, say O_p and O_q where $1 \leq p, q \leq W$, and are moved away from the centroid of X_i and X_j by a constant Δ_2.

Asymptotically, the algorithm groups the objects into partitions based on the nearness of X_i's. The BAM can be viewed as a clustering algorithm that moves the requested pairs towards each other as opposed to the means or medoids of the object clusters. Unfortunately, the BAM takes a long time to converge and also makes use of floating-point numbers, which are undesirable for quick computations. The algorithmic description of the BAM is presented in Algorithm 1.

2.3.3 Tsetlin and Krinsky Strategies to the OPP

Two FSSA solutions were proposed in [37, 47] to solve the EPP. This techniques made use of the Tsetlin and Krinsky automata to partition the object into groups.

As we saw earlier, the Tsetlin $L_{KN,K}$ automaton has K actions, $\{\alpha_1, \ldots, \alpha_K\}$ and KN states $\{\phi_1, \ldots, \phi_{KN}\}$, where N is state-depth of the automaton. The states of the automaton are separated into K groups, each with a state depth of N. For groups $i, 2, ..., K$, when $i \neq \{1, K\}$, the states range from $(i-1)N+1$ through to iN. When $i = 1$, the states go from 1 to N, and when $i = k$, the states span $(N-1)K+1$ to KN. Figure 2.8 presents a graphical representation of this multi-action automaton. The reader will observe that the boundary states $N, ...iN, ..., KN$ all lie at the center of the state diagram, while the states $1, ...(i-1)N+1, ..., (N-1)K+1$ are the innermost or internal states of their respective actions or groups.

When an action is rewarded, the automaton moves a step towards the innermost state of the action. However, if the automaton is in states $1, ...(i-1)N+1, ..., (N-1)K+1$, it remains there. On the other hand, on receiving a penalty, the automaton moves a step towards the boundary states of the automaton, and if the automaton is in states $N, ...iN, ..., KN$, it then moves to the boundary-state of the subsequent action (mod K). So, $\forall\ 1 \leq i \leq K$,

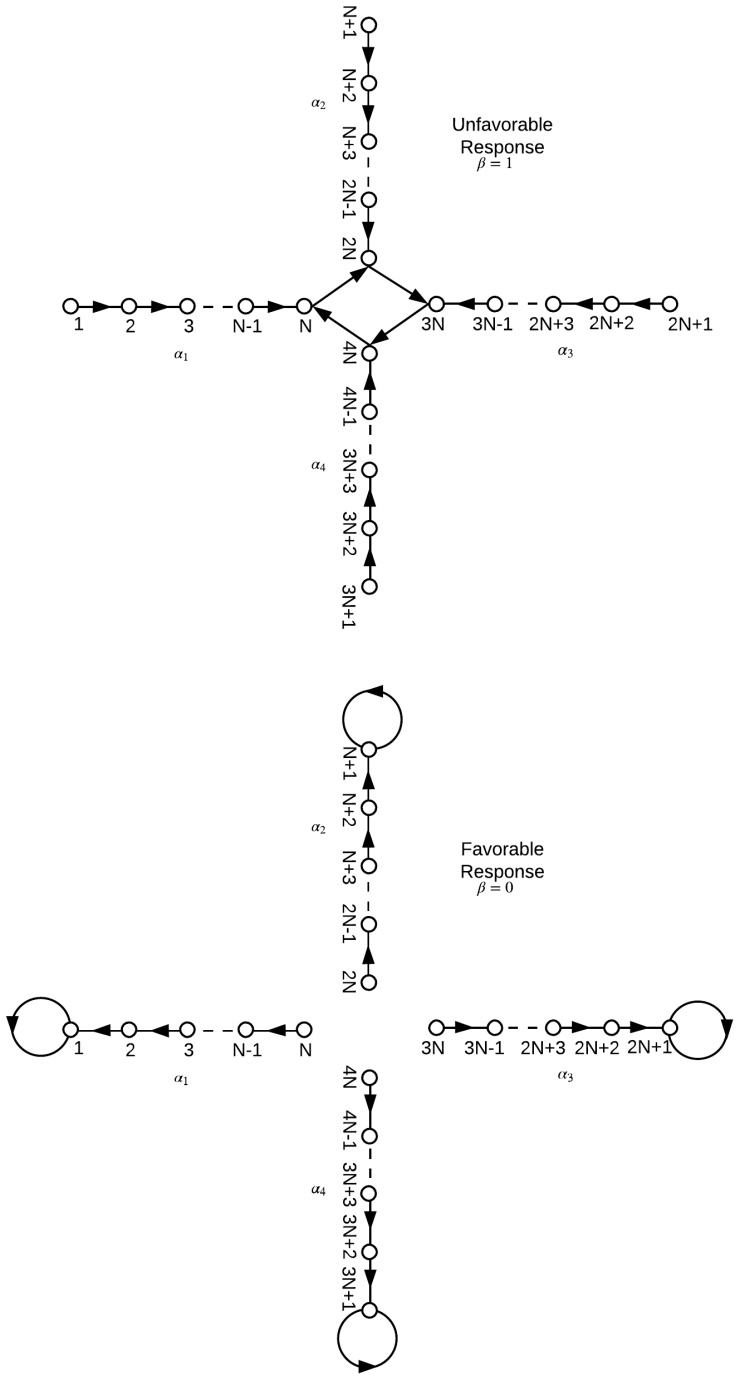


Figure 2.8: The Multi-action transition graph for the Tsetlin L_{KN}, K automaton ($K = 4$).

$$\begin{aligned} F_{\text{Tsetlin}}(\phi_j, 0) &= \phi_{j-1} && \text{if } (i-1)N+1 < j \leq iN \\ &= \phi_j && \text{if } j = (i-1)N+1, \end{aligned}$$

and,

$$\begin{aligned} F_{\text{Tsetlin}}(\phi_j, 1) &= \phi_{j+1} && \text{if } (i-1)N+1 \leq j < iN \\ &= \phi_{((i+1)N) \bmod KN} && \text{if } j = iN. \end{aligned}$$

Krinsky's automaton, on the other hand, is identical to Tsetlin's automaton on receiving a penalty. However, on being rewarded, it moves directly to the innermost state of the action it was rewarded on. Hence the transition rule on reward is:

$$F_{\text{Krinsky}}(\phi_j, 0) = \phi_{(i-1)N+1} \qquad \text{if } (i-1)N+1 \leq j \leq iN.$$

Observe that the actions of the automaton correspond to the different positions of the abstract objects. For the LA, if an abstract object is in a given action α_k at time t, and if it receives at instant $t+1$, a pair of abstract query elements $\langle O_i, O_j \rangle$, the automaton behaves as follows:

- If the pair $\langle O_i, O_j \rangle$ are found to be in the same group, the LA is rewarded, and hence it moves according to the corresponding reward-update rule of the scheme in operation (i.e., either Tsetlin or Krinsky).
- Further, if $\langle O_i, O_j \rangle$ are not grouped together, the automaton is penalized and the abstract objects are moved towards the boundary state of α_k, But if it already is in a boundary state, the automaton changes to the boundary state of another action, thereby changing the grouping.

From the results reported in [47], the Tsetlin and Krinsky solution to the OPP/ EPP problem yielded good results compared to the BAM. Krinsky's automaton performed even better when compared to the BAM. As before, for a given element O_i in partition π_k, the query followed the distribution:

$$\sum_{A_j \in \pi_k} Pr\{A_i, A_j \text{ accessed together}\} = p,$$

where p is the joint access probability. When $p = 0.9$ and when the number of objects, $W = 4$, and the number of groups or classes, $R = 2$, the BAM method required 80 iterations, while the Tsetlin and Krinsky needed 14 and 7 iterations respectively to converge to the optimal solution. Also, when $W = 6$ (for the same value of R), it took the BAM, the Tsetlin and the Krinsky LA, 240, 20 and 12 iterations respectively. These results clearly highlighted the superiority of the LA-based FSSA over the BAM.

That being said, a major drawback of the Tsetlin and Krinsky schemes is the impractical number of *actions* required to partition the objects when W and R exceeded the small values used in the experiments reported in [47]. The number of actions needed can be seen to be $\frac{W!}{((W/R)!)^R R!}$, rendering this method to be undesirable for many practical systems.

2.3.4 The Object Migration Automaton (OMA)

The Object Migration Automaton or OMA improves on the Tsetlin/ Krinsky strategies for partitioning objects into groups in the EPP. In the OMA, the number of actions represents the number of groups, or partitions, R, where each action contains a certain number of states, N. The OMA partitions the object set W into groups R by moving the abstract objects $\mathcal{O}$ around the action-states of the automaton. This strategy is unlike the Tsetlin and Krinsky methods that instead moves the entire set of objects towards an action. So, for example, if object O_i is in action α_k, then O_i belongs to the k^{th} group.

The state-action setup of the OMA is similar to Tsetlin automaton where an action α_k has N states $\{\phi_{k1}, \phi_{k2}, ..., \phi_{kN}\}$ where ϕ_{k1} is the innermost state and ϕ_{kN} is the boundary state. The automaton receives queries in the form of tuples, $\langle A_i, A_j \rangle$. Hence, given a set of query elements, we formalize the transitions of the OMA for the abstract objects $\langle O_i, O_j \rangle$ on reward and on penalty as follows:

- **On reward**: If the query objects $\langle O_i, O_j \rangle$ are in the same class (or group, or

action), the automaton is rewarded, and the objects are moved towards the innermost state of the action, ϕ_{k1}.

- **On penalty**: If the query objects $\langle O_i, O_j \rangle$ are **not** in the same class, the automaton is penalized, and the objects are moved towards the boundary states ϕ_{Nk} of their respective classes. The movement on penalty depends on the states that the objects pairs $\langle O_i, O_j \rangle$ are at:
  - ***If both objects,*** $\langle O_i, O_j \rangle$ ***are NOT at boundary states***: Move O_i and O_j one step towards the boundary state of their respective actions. Formally, if $O_i \neq \phi_{mN}$ and $O_j \neq \phi_{kN}$, move O_i towards ϕ_{mN} and O_j towards ϕ_{kN}.
  - ***If one object,*** O_i ***or*** O_j ***is at a boundary state***: In this case, either $O_i = \phi_{mN}$ or $O_j = \phi_{kN}$, but not both.
    - Assuming $O_i = \phi_{mN}$, move O_j a step towards its boundary state ϕ_{kN} and move O_i to the boundary state of O_j. In doing so, we observe that the class ϕ_k now has one extra element resulting in an imbalanced class-set. So, to balance the classes, take the closest object to ϕ_{kN} that is not O_i or O_j and move it to the boundary state ϕ_{mN}.
    - But if $O_j = \phi_{kN}$, move O_i a step towards its boundary state ϕ_{mN} and move O_j to the boundary state of O_i. Then take the closest object to ϕ_{mN} that is not O_i or O_j and move it to the boundary state ϕ_{kN}.
  - ***If both objects*** $\langle O_i, O_j \rangle$ ***are on boundary states***: In this case, both $O_i = \phi_{mN}$ and $O_j = \phi_{kN}$. Hence, there are two choices for a move and the program can execute either of them.
    - Choice 1: When $O_i = \phi_{mN}$, move O_j to the boundary state of O_i. Note that this will result in an unbalanced class for ϕ_{mN}. To remedy this, take the closest object to ϕ_{mN} that is not O_i or O_j and move it to the boundary state ϕ_{kN}
    - Choice 2: When $O_j = \phi_{kN}$, move O_i to the boundary state of O_j. Note that this will result in an unbalanced class for ϕ_{kN}. To remedy

this, take the closest object to ϕ_{kN} that is not O_i or O_j and move it to the boundary state ϕ_{mN}

The move sequence for the OMA outlined above are graphically represented in Figure 2.9. The algorithmic descriptions are also presented in Algorithms 2, 3, and 4.

Algorithm 2 The OMA 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:

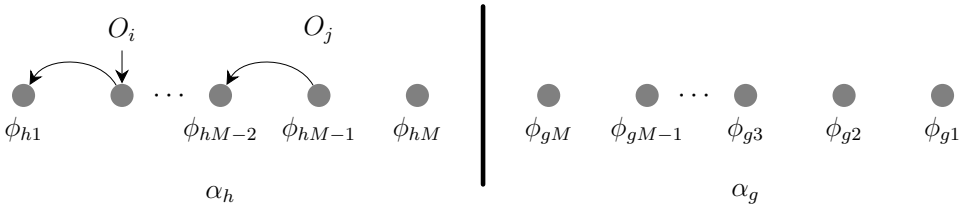
- A periodic clustering of the objects into R partitions.
- ξ_i is the state of the abstract object O_i. It is an integer in the range $\{1, 2, \cdots R.N\}$, where, if $(k-1)N + 1 \leq \xi_i \leq kN$ then object O_i is assigned to α_k.

```
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 ProcessReward({ξ_p}, A_i, A_j)
7:         else                                   ▷ The partitioning is penalized
8:             Call ProcessPenalty({ξ_p}, A_i, A_j)
9:         end if
10:    end for
11:    Decide on the final partitions based on the states {ξ_i}
12: end
```

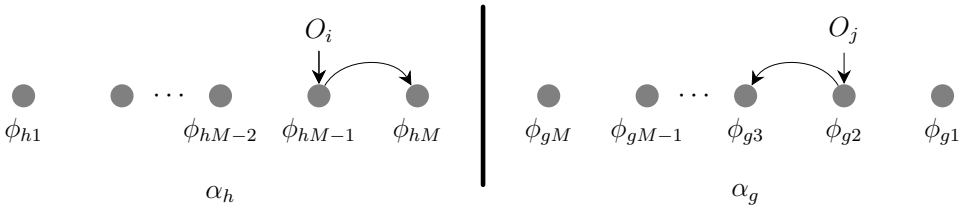
Source: Excerpt from [55]

Performance Analysis

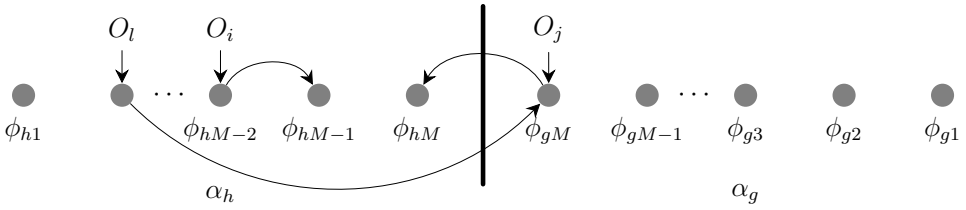
The OMA has been experimentally proven to perform better than the BAM strategy [47]. Consider a set of elements in which $W = 18$, and the number of classes, $R = 2$. For the probability of the elements being in the same group, $p = 0.9$, the BAM required 3352 iterations, while OMA required only 277. And as the number of



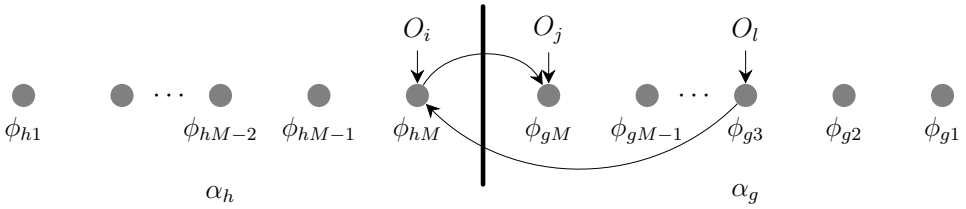
(a) On reward: Move the accessed abstract objects $\langle O_i, O_j \rangle$ towards the extreme states.



(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 2.9: The OMA Algorithm
Source: Excerpt from [55]

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

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 [55]

objects to be partitioned increased the convergence rate of the OMA was an order of magnitude superior to the BAM. Although the convergence rates for Tsetlin's and Krinsky's automata are better than the OMA, it has the advantage of being computationally feasible to partition large enough object sets into classes. The OMA's performance appeared to be ϵ-optimal from its empirical performance. A formal analysis does not exist, although, when $W = 2$, and $R = 2$ it is proven to be, at least, expedient.

Critical Implementation Strategies

In implementing the OMA, the W objects for partitioning are distributed evenly on the boundary states of the given groups, which correspond to the actions of the OMA [3]. This is implemented by using the rule:

$$(((i-1) \bmod R) + 1) \times N,$$

for all $1 \leq i \leq W$. In doing so, each action α_k has exactly $\frac{W}{R}$ objects. Also, the class that a particular object belongs to can be derived by the rule:

$$((\mathcal{O}[i]-1) \operatorname{div} N) + 1,$$

Algorithm 4 *ProcessPenalty*$(\{\xi_p\}, A_i, A_j)$

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
    $\xi_i = \xi_i + 1$
  else if $\xi_j \bmod N \neq 0$ **then** ▷ O_j is at internal state
    $\xi_j = \xi_j + 1$
  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
    $\xi_l = temp$ ▷ Move O_l to the old state of O_i
  end if
end

Source: Excerpt from [55]

where $\mathcal{O}$ is the set of abstract objects.

OMA Convergence

In quantifying the convergence of the OMA, two index metrics are recorded while executing the partitioning algorithm. The indices are reported as a tuple $\langle t_1, t_2 \rangle$ where t_1 is the first time instant when all the objects settle in its optimal partition. Further, t_2 is the time index at which all the abstract objects are at the most internal states of their respective classes (or actions). These two metrics are vital in reporting the convergence and performance capabilities of the OMA for various Environments. Table 2.1 below shows the empirical convergence results for the OMA (as reported in [55]) for different sizes of W and R, with $p = 0.7$, 0.8 and 0.9. Note that $p = 0.7$ is the most difficult of these Environments.

Table 2.1: Experimental results for the OMA done for an ensemble of 100 experiments in which we have only included the results from experiments where convergence has occurred.

W	W/R	R	OMAp9	OMAp8	OMAp7
4	2	2	$(2, 26)$	$(2, 36)$	$(2, 57)$
6	2	3	$(3, 44)$	$(4, 62)$	$(4, 109)$
-	3	3	$(22, 66)$	$(20, 88)$	$(26, 153)$
9	3	3	$(44, 110)$	$(43, 144)$	$(70, 261)$
12	2	6	$(10, 101)$	$(12, 146)$	$(15, 285)$
-	3	4	$(82, 172)$	$(84, 228)$	$(128, 406)$
-	4	3	$(401, 524)$	$(252, 405)$	$(256, 552)$
-	6	2	$(2240, 2370)$	$(1151, 1299)$	$(1053, 1486)$
15	3	5	$(152, 265)$	$(155, 325)$	$(191, 607)$
-	5	3	$(1854, 2087)$	$(918, 1136)$	$(735, 1171)$
18	2	9	$(17, 167)$	$(24, 252)$	$(29, 582)$
-	3	6	$(180, 319)$	$(202, 413)$	$(288, 839)$
-	6	3	$(5660, 5786)$	$(1911, 2265)$	$(1355, 2111)$
-	9	2	$(11245, 11456)$	$(6494, 7016)$	$(3801, 4450)$

The results are given as a pair (a, b) where a refers to the number of iterations for the OMA to reach the first correct classification and b refers to the case where the OMA has fully converged.
In all experiments, the number of states of the OMA is set to 10.
OMAp$\mathcal{X}$: $\mathcal{X}$ refers to the Environment's probability of generating samples within the same class.
N: Number of objects to be partitioned.
W/R: Number of objects in every class.
R: Number of classes in the partitioning problem.
Source: Excerpt from [55]

2.3.5 The Stochastic Move Automaton (SMA)

The SMA or Stochastic Move Automaton was proposed by the authors of [46] to mitigate the known short-comings of the OMA, which is particularly well suited for solving the EPP, and which hence, cannot be extended as a more generalized partitioning algorithm. Also, the OMA possesses the drawback that it can sometimes

place an object permanently in a class that it should not be in. The SMA is, overall, very similar to the OMA in implementation. However, it extends the OMA with the following rules:

- The number of actions, correspond to the size of the object list, W. So, every abstract object is, at inception, placed in its corresponding class.
- A variable number of objects can be in the same class. The objects do not have to be equipartitioned.
- If one of the query pairs is at a boundary state, it can easily move to another action.
- The decision to move from a boundary state to another class is made with a probability, δ. Hence, an object stays with δ in the same class and moves to a different action with probability $1 - \delta$.

The algorithmic descriptions for the SMA are presented in Algorithm 5, 6 and 7.

2.3.6 The Modified Linear Reward Penalty Scheme (ML_{RP})

The Modified Linear Reward Penalty (ML_{RP}) scheme is a variable-structure LA solution to the OPP. It was proposed to further mitigate the problem of outliers or rare event queries that can "upset" the optimal partitioning of the set of abstract objects, $\mathcal{O}$.

The ML_{RP} uses the Linear Reward-Penalty reinforcement scheme provided to decide on the probability distribution for the partitioning of the different objects. It possesses W actions, $\{\alpha_1, ..., \alpha_W\}$, where each action corresponds to an abstract object being in $\{O_1, ..., O_W\}$. Thus, each abstract object can probabilistically choose its respective action based on the ML_{RP} scheme.

The probability distribution is represented as a set of probability vectors P, where $P = \{\underline{P}_1, ..., \underline{P}_2\}$, and $\underline{P}_i = [p_{i1}, ..., p_{iW}]^T$ is a probability vector containing the probabilities, p_{ij} that an abstract object O_i chooses a particular action $\{\alpha_i, ..., \alpha_W\}$.

Algorithm 5 SMA 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.
- ξ_i is the state of the abstract object O_i. It is an integer in the range $\{1, 2, \cdots R.N\}$, where, if $(k-1)N + 1 \leq \xi_i \leq kN$ then object O_i is assigned to α_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 SMAProcessReward({ξ_p}, A_i, A_j)
7:         else                                   ▷ The partitioning is penalized
8:             Call SMAProcessPenalty({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
```

Algorithm 6 *SMAProcessReward*$(\{\xi_p\}, A_i, A_j)$

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
```

Algorithm 7 $SMAProcessPenalty(\{\xi_p\}, A_i, A_j)$

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 ≠ 0 ∧ ξ_j mod N ≠ 0 then          ▷ Both are in internal states
3:         ξ_i = ξ_i + 1
4:         ξ_j = ξ_j + 1
5:     else if ξ_i mod N ≠ 0 then                     ▷ O_i is at internal state
6:         ξ_i = ξ_i + 1
7:         if (RAND < δ) then ξ_j = ⌈ξ_i/N⌉ * N▷ Move O_j to O_i's class with prob δ
8:         end if
9:     else if ξ_j mod N ≠ 0 then                     ▷ O_j is at internal state
10:        ξ_j = ξ_j + 1
11:        if (RAND < δ) then ξ_i = ⌈ξ_j/N⌉ * N ▷ Move O_i to O_j's class with prob
   δ
12:        end if
13:    else                                           ▷ Both are in boundary states
14:        temp1 = ξ_i                                ▷ Store the state of O_i
15:        temp2 = ξ_j                                ▷ Store the state of O_j
16:        if (RAND < δ) then ξ_i = temp2
17:        end if
18:        if (RAND < δ) then ξ_j = temp1
19:        end if
20:    end if
21: end
```

For a pair of queried objects $\langle A_i, A_j \rangle$ if O_i and O_j choose the same action α_k, they are rewarded, and the corresponding probability p_{ik} and p_{jk} of their respective probability vectors are increased. Moreover, the probabilities of other actions are decreased.

On the other hand, if O_i and O_j choose different actions, α_p and α_q, the probabilities are updated in such a way that the objects get closer to choosing the same action. For O_i, the probability p_{ip} is decreased, while the probability p_{iq} is increased. Analogously, for O_j, the probability p_{jq} is decreased and the probability p_{jp} increased.

In both cases, the probability of choosing the other actions are left untouched. The algorithm is provided in Algorithm 8.

Algorithm 8 ML_{RP} Algorithm

Initialization:

- Initialize p_{ii} to 1.0 and p_{ij} to 0.0

begin
  Initialization of $\{\xi_p\}$
  for a sequence of T queries **do**
    Read query $\langle A_i, A_j \rangle$
    if $(\alpha_p = m)$ and $(\alpha_q = m)$ **then** ▷ Reward: Both objects chose same class
      $p_{im} = 1 - \lambda_1(1 - p_{im})$ ▷ increase chance to choose α_p
      $p_{jm} = 1 - \lambda_1(1 - p_{jm})$ ▷ increase chance to choose α_q
      for $k = 1$ to W, $k \neq m$ **do**
        $p_{ik} = \lambda_1 p_{ik}$ ▷ decrease chance to choose
        $p_{jk} = \lambda_1 p_{jk}$ ▷ other actions
      end for
    else if $(\alpha_p = m)$ and $(\alpha_q = n)$ **then** ▷ Penalty: Different actions chosen
      $p_{in} = p_{in} + (1 - \lambda_2)p_{im}$ ▷ Encourage O_i to choose α_q
      $p_{im} = \lambda_2 p_{im}$ ▷ and not α_p
      $p_{jm} = p_{jm} + (1 - \lambda_2)p_{jn}$ ▷ Encourage O_j to choose α_p
      $p_{jn} = \lambda_2 p_{jn}$ ▷ and not α_q
    end if
  end for
  Present partitions based on the the probabilities p_{ij}'s
end

2.3.7 The *Pursuit*-OMA (POMA)

The Pursuit philosophy, introduced earlier for the field of LA, has been incorporated into the OMA to mitigate the adverse effects of bad/divergent queries (which occurs as a consequence of working with noisy Environments or a large number of objects in each group) that degrade the performance of the OMA [55]. The Pursuit concept improves the OMA in two unique ways.

The Pursuit strategy is employed to filter out divergent queries from the Environment in real-time *before* the queries are passed on to the partitioning algorithm. This

paradigm uses "cheap" (or rather "inexpensive") estimates of the reward probabilities to approximate the queries coming from the Environment. This strategy differs from the vanilla OMA approach which uses the query pairs to partition objects but does not take into consideration the frequencies of the query pairs. As a consequence of this, the pursuit concept upgrades the vanilla OMA by obtaining information on the frequency of a query pair, and uses this information to approximate or estimate the "certainty" of that pair. This certainty is controlled by a pre-defined parameter, τ. If the frequency of a query pair is less than τ, it is seen to be a divergent pair.

Thus, apart from processing the set of incoming queries to reject divergent pairs, the pursuit strategy also updates the action-probability vector by ranking the estimates of the reward probabilities (in addition to the response of the Environment). Consequently, the superior actions are chosen more often as the reward estimates get more accurate. This technique is shown to speed up the convergence of the algorithm. The algorithm, referred to as the Pursuit OMA (POMA) is presented in Algorithm 9.

Performance of POMA

The empirical convergence of the POMA is again estimated (as in the case of the OMA) by recording a tuple of indices. The first index is when all the abstract objects settle in their optimal partition, and the second is when all the abstract objects are in the most internal states of their respective groups. Table 2.2 below shows the convergence rates for the POMA as reported in [55]. In the experiments, the threshold τ for the first two rows was 0.01, and 0.02 for the rest of the rows. Moreover, the iterations needed for estimating τ was 10 for rows 1 and 2, 60 for rows 3, 4 and 5, and 120 for the last row. The table displays the convergence results for different sizes of W and R, with p = 0.7, 0.8 and 0.9. From Table 2.2, it is easy to observe that the results of POMA are two to three times superior to that of OMA. For a detailed discussion of the results, please refer to [55].

Algorithm 9 POMA algorithm

Input:

- A matrix of counters to yield frequencies, $\mathcal{Z}$, initially set to zeros, whence $\mathcal{P}$ is computed.
- A user-defined threshold, τ, set to a value reasonable close to zero.
- The abstract objects $\{O_1, \cdots, O_W\}$.
- The number of states N per action.
- A stream of queries $\langle A_i, A_j \rangle$.
- κ is the number of iterations required for $\mathcal{P}$ to "converge".

Output:

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

1: **begin**
2: The initialization of $\{\xi_p\}$, as described in the text.
3: **for** a sequence of $i = 1, \cdots, T$ queries **do**
4: Read query $\langle A_i, A_j \rangle$
5: $\mathcal{Z}[A_i, A_j] = \mathcal{Z}[A_i, A_j] + 1$
6: $\mathcal{Z}[A_j, A_i] = \mathcal{Z}[A_i, A_j]$
7: $\mathcal{P}[A_i, A_j] = \dfrac{\mathcal{Z}[A_i, A_j]}{\sum_{k,l=1}^{W} Z[A_k, A_l]}$ ▷ Update the statistics
8: **if** $i < \kappa$ **then** ▷ Statistics have not converged; Invoke OMA
9: **if** ξ_i div $N = \xi_j$ div N **then** ▷ The partitioning is rewarded
10: Call ProcessReward($\{\xi_p\}, A_i, A_j$)
11: **else** ▷ The partitioning is penalized
12: Call ProcessPenalty($\{\xi_p\}, A_i, A_j$)
13: **end if**
14: **else if** $\mathcal{P}[A_i, A_j] \geq \tau$ **then** ▷ Statistics have converged
15: **if** ξ_i div $N = \xi_j$ div N **then** ▷ Reward partitioning if $\mathcal{P}[A_i, A_j] \geq \tau$
16: Call ProcessReward($\{\xi_p\}, A_i, A_j$)
17: **else** ▷ Penalize partitioning if $\mathcal{P}[A_i, A_j] \geq \tau$
18: Call ProcessPenalty($\{\xi_p\}, A_i, A_j$)
19: **end if**
20: **else** ▷ Filter the Divergent queries
21: **end if**
22: **end for**
23: Print out the partitions based on the states $\{\xi_i\}$
24: **end**

Source: Excerpt from [55]

Table 2.2: Experimental results for the POMA done for an ensemble of 100 experiments in which we have only included the results from experiments where convergence has occurred.

W	W/R	R	POMAp9	POMAp8	POMAp7	τ	κ
4	2	2	$(2, 24)$	$(2, 28)$	$(2, 36)$	$1.5\tau^*$	κ^*
6	2	3	$(3, 35)$	$(4, 46)$	$(4, 61)$	$1.5\tau^*$	κ^*
-	3	2	$(20, 60)$	$(20, 75)$	$(31, 107)$	τ^*	$(W/R)\kappa^*$
9	3	3	$(44, 107)$	$(61, 136)$	$(76, 171)$	τ^*	$(W/R)\kappa^*$
12	2	6	$(10, 96)$	$(12, 124)$	$(15, 165)$	τ^*	$(W/R)\kappa^*$
-	3	4	$(84, 166)$	$(101, 201)$	$(112, 244)$	τ^*	$(W/R)\kappa^*$
-	4	3	$(236, 339)$	$(231, 373)$	$(261, 486)$	$0.1\tau^*$	$(W/R)\kappa^*$
-	6	2	$(1609, 1739)$	$(947, 1169)$	$(898, 1269)$	$0.1\tau^*$	$(W/R)\kappa^*$
15	3	5	$(142, 239)$	$(155, 292)$	$(195, 356)$	τ^*	$(W/R)\kappa^*$
-	5	3	$(1658, 1879)$	$(786, 972)$	$(599, 1054)$	$0.1\tau^*$	$(W/R)\kappa^*$
18	2	9	$(17, 163)$	$(24, 219)$	$(29, 295)$	τ^*	$(W/R)\kappa^*$
-	3	6	$(182, 302)$	$(196, 350)$	$(306, 494)$	τ^*	$(W/R)\kappa^*$
-	6	3	$(5434, 5563)$	$(1503, 1803)$	$(1263, 1881)$	$0.1\tau^*$	$(W/R)\kappa^*$
-	9	2	$(9754, 10436)$	$(5075, 5522)$	$(3801, 4544)$	$0.1\tau^*$	$(W/R)\kappa^*$

The results are given as a pair (a, b) where a refers to the number of iterations for the POMA to reach the first correct classification and b refers to the case where the POMA has fully converged.
In all experiments, the number of states of the POMA is set to 10.
POMAp$\mathcal{X}$: $\mathcal{X}$ refers to the Environment's probability of generating samples within the same class, i.e., POMAp9 means $p = 0.9$.
N: Number of objects to be partitioned.
W/R: Number of objects in every class.
R: Number of classes in the partitioning problem.
The default entry in the second-last column for τ is $\tau^* = \frac{1}{W^2}$. Otherwise, the value of τ is specified in each row.
The default entry in the last column for κ is $\kappa^* = R\left[(\frac{W}{R})^2 - \frac{W}{R}\right]$. Otherwise, the value of κ is specified in each row.
Source: Excerpt from [55]

2.3.8 Enhanced-OMA (EOMA) / *Pursuit*-EOMA (PEOMA)

The "Enhanced" OMA (EOMA) is an upgraded embodiment of the OMA algorithm proposed by the authors of [21] to mitigate the susceptibility of the OMA algorithm

to a "deadlock situation" which prevents the algorithm from converging to the objects' optimal partitioning. The deadlock condition is actually 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.

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, α_m and α_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. Otherwise, the update is identical to the OMA.

The EOMA also redefines the concept of the convergence condition so as to reduce the algorithms 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. For the analysis on the EOMA, the reader is referred to [21, 55]. The algorithmic description for the EOMA is shown in Algorithms 10, 11 and 12.

The empirical performance of the EOMA, as recorded in [55], is shown in Table 2.3. Observe that as the difficulty of the Environment increases, the convergence rate is superior to that of the OMA. For a detailed discussion of the performance of the EOMA, the reader is referred to [55].

PEOMA

The pursuit concept has also been incorporated into the EOMA to enhance it to yield the Pursuit EOMA (PEOMA). This scheme was the best-known object-partitioning algorithm until the authors [56] introduced the TPEOMA. Again, the pursuit concept, just as in the case of the POMA, filters queries by rejecting divergent queries. It updates the joint query probabilities by using the ranked estimates of the reward

Algorithm 10 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.
- ξ_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 α_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 [55]

probabilities to asymptotically choose or "pursue" better actions. The deadlock phenomenon is, of course, mitigated by using the EOMA instead of the OMA. For a detailed discussion on how the pursuit concept is incorporated in the design, the reader is referred to [55]. The algorithmic description of the PEOMA is given in Algorithm 13.

The convergence rate of the PEOMA algorithm is 25% faster than that of the EOMA, and over 2.5 times faster than that of POMA. The empirical results of the PEOMA in various Environments and object-group configurations (as reported in [55]) is shown in Table 2.4. As before, detailed discussions of the performances of the PEOMA, EOMA, and POMA are given in [55].

Algorithm 11 *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 [55]

2.3.9 Transitivity for PEOMA - (TPEOMA)

A further enhancement to the PEOMA was proposed in the paper [56]. This enhancement is based on the observation that the Pursuit matrix can also be used to infer underlying relations in the Environment. It then invokes a policy that spins off reward/penalty operations by incorporating the statistics obtained between objects

Algorithm 12 *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 [55]

that have been previously accessed.

In fact, this paradigm seeks for the transitivity relation between the objects. When an estimate of the Pursuit matrix is obtained, the transitivity property can be *inferred*. In other words, if the current query received from the Environment is $\langle O_i, O_j \rangle$ and O_j is in a known relation with $O_k, k \in \{1, \cdots, W_j\}$, $k \neq i$, one can infer that O_i is also in such a relation with O_k, where i, j, and k are the indices of the entries in the Pursuit matrix. Thus, if τ_T is a suitable user-defined threshold, one can assert that $P_{ij} > \tau_T \wedge P_{jk} > \tau_T \Rightarrow P_{ik} > \tau_T$. This means that if i and j, and i and k are also accessed together often, it is also likely that i and k are also accessed together often.

The reader is referred to [56] for the details on how to incorporate transitivity and the Pursuit paradigm into the OMA. However, suffice it to say that the transitive implementation of the PEOMA has, in some cases, about a 100% increase in the convergence rate, as opposed to the non-transitive PEOMA implementations. The TPEOMA is currently, at the time of writing, the most efficient solution for the EPP. The algorithmic description for the TPEOMA is given in Algorithm 14.

Table 2.5 depicts the performance of the TPEOMA. For a detailed discussion of the corresponding simulation results of the TPEOMA in comparison with the

Table 2.3: Experimental results for the *Enhanced* OMA (EOMA) done for an ensemble of 100 runs.

W	W/R	R	EOMAp9	EOMAp8	EOMAp7
4	2	2	$(2, 26)$	$(2, 30)$	$(3, 60)$
6	2	3	$(4, 46)$	$(4, 65)$	$(5, 106)$
-	3	2	$(6, 50)$	$(8, 74)$	$(11, 127)$
8	2	4	$(6, 64)$	$(7, 95)$	$(8, 158)$
-	4	2	$(14, 75)$	$(20, 110)$	$(32, 185)$
9	3	3	$(18, 91)$	$(24, 132)$	$(35, 233)$
10	5	2	$(8, 85)$	$(10, 118)$	$(13, 226)$
-	2	5	$(25, 106)$	$(33, 153)$	$(70, 277)$
12	2	6	$(10, 102)$	$(12, 154)$	$(17, 291)$
-	3	4	$(43, 136)$	$(56, 207)$	$(81, 380)$
-	4	3	$(54, 150)$	$(66, 196)$	$(99, 388)$
-	6	2	$(40, 133)$	$(64, 208)$	$(105, 405)$
15	3	5	$(65, 187)$	$(92, 284)$	$(134, 554)$
-	5	3	$(75, 191)$	$(108, 295)$	$(192, 617)$
18	2	9	$(19, 170)$	$(26, 253)$	$(36, 630)$
-	3	6	$(106, 258)$	$(140, 389)$	$(242, 827)$
-	6	3	$(114, 255)$	$(167, 392)$	$(261, 857)$
-	9	2	$(112, 246)$	$(142, 363)$	$(311, 854)$

The results are given as a pair (a, b) where a refers to the number of iterations for the EOMA to reach the first correct classification and b refers to the case where the EOMA has fully converged.
In all experiments, the number of states of the EOMA is set to 10.
EOMAp$\mathcal{X}$: $\mathcal{X}$ refers to the Environment's probability of generating samples within the same class.
N: Number of objects to be partitioned.
W/R: Number of objects in every class.
R: Number of classes in the partitioning problem.
Source: Excerpt from [55]

PEOMA and with the original OMA/EOMA, the reader is referred to [56]. However for the case mentioned earlier when $W = 18$ and $R = 3$ with $p = 0.7$, the PEOMA required 472 iterations, while the TPEOMA required only 244 iterations.

Algorithm 13 PEOMA algorithm

Input:

- A matrix of frequencies, $\mathcal{P}$, initially set to zeros.
- A user-defined threshold, τ, set to a value reasonable close to zero.
- The abstract objects $\{O_1, \cdots, O_W\}$
- The number of states N per action
- A stream of queries (A_i, A_j)

Output:

- A periodic clustering of the objects into R partitions.
- ξ_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 α_k.

begin
  Initialization of $\{\xi_p\}$ ▷ This is described in the text
  for a sequence of T queries **do**
    Read query (A_i, A_j)
    $\mathcal{P}[A_i, A_j] = \mathcal{P}[A_i, A_j] + 1$ ▷ Update the statistics
    if $i < \kappa$ **then** ▷ Invoke the EOMA
      $SkeletalEOMA(\{\xi_p\}, A_i, A_j)$
    else
      if $\mathcal{P}[i, j] / \sum_{i=1}^{W} \mathcal{P}[i, j] \geq \tau$ **then** ▷ Filter the Divergent queries
        if ξ_i div $N = \xi_j$ div N **then** ▷ The partitioning is rewarded
          Call ProcessRewardEOMA($\{\xi_p\}, A_i, A_j$)
        else ▷ The partitioning is penalized
          Call ProcessPenaltyEOMA($\{\xi_p\}, A_i, A_j$)
        end if
      end if
    end if
  end for
  Print out the partitions based on the states $\{\xi_i\}$
end

Source: Excerpt from [55]

2.4 Adaptive Lists

This section examines the existing research from the literature on self-organizing lists.

Algorithm 14 TPEOMA

Input:

- A matrix of counters to yield frequencies, $\mathcal{Z}$, initially set to zeros, whence $\mathcal{P}$ is computed.
- A user-defined threshold, τ, set to a value reasonable close to zero.
- A user-defined threshold, τ_t, set to a value greater than $\frac{1}{W^2 - W}$.
- The number of states N per action.
- A stream of queries $\langle A_i, A_j \rangle$.
- K is the number of iterations required for $\mathcal{P}$ to "converge".

Output:

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

1: **begin**
2: The initialization of $\{\xi_p\}$, as described in the text.
3: **for** a sequence of $i \leftarrow 1, \cdots, T$ queries **do**
4: Read query $\langle A_i, A_j \rangle$
5: $\mathcal{Z}[A_i, A_j] \leftarrow \mathcal{Z}[A_i, A_j] + 1$
6: $\mathcal{Z}[A_j, A_i] \leftarrow \mathcal{Z}[A_i, A_j]$
7: $\mathcal{P}_{i,j} \leftarrow \frac{\mathcal{Z}[A_i, A_j]}{\sum_{k,l=1}^{W} Z[A_k, A_l]}$ ▷ Update stats
8: **if** $i < \kappa$ **then** ▷ Invoke the EOMA
9: $SkeletalEOMA(\{\xi_p\}, A_i, A_j)$
10: **else**
11: **if** $\mathcal{P}_{ij} > \tau$ **then** ▷ Valid query
12: $SkeletalEOMA(\{\xi_p\}, A_i, A_j)$
13: **for** $l \leftarrow \{1, \cdots, W\} \wedge l \neq i, j$ **do**
14: **if** $\mathcal{P}_{il} > \tau_t$ **then**
15: $SkeletalEOMA(\{\xi_p\}, A_i, A_l)$
16: **end if**
17: **end for**
18: **end if**
19: **end if**
20: **end for**
21: Print out the partitions based on the states $\{\xi_i\}$
22: **end**

Source: Excerpt from [56]

Table 2.4: Experimental results for the PEOMA approach done for an ensemble of 100 runs.

W	W/R	R	PEOMAp9	PEOMAp8	PEOMAp7
4	2	2	$(2, 23)$	$(2, 37)$	$(3, 44)$
6	2	3	$(4, 42)$	$(4, 52)$	$(5, 73)$
-	3	2	$(7, 47)$	$(8, 62)$	$(10, 91)$
8	2	4	$(6, 59)$	$(6, 76)$	$(8, 102)$
-	4	2	$(15, 73)$	$(23, 100)$	$(36, 145)$
9	3	3	$(20, 85)$	$(24, 110)$	$(40, 146)$
10	2	5	$(8, 79)$	$(10, 102)$	$(12, 141)$
-	5	2	$(26, 100)$	$(36, 140)$	$(54, 213)$
12	2	6	$(10, 97)$	$(12, 129)$	$(17, 181)$
-	3	4	$(38, 126)$	$(55, 165)$	$(74, 222)$
-	4	3	$(44, 134)$	$(58, 165)$	$(87, 241)$
-	6	2	$(34, 127)$	$(60, 182)$	$(110, 310)$
15	3	5	$(72, 174)$	$(88, 228)$	$(147, 308)$
-	5	3	$(76, 185)$	$(105, 249)$	$(155, 348)$
18	2	9	$(19, 166)$	$(26, 218)$	$(36, 323)$
-	3	6	$(98, 231)$	$(139, 310)$	$(207, 419)$
-	6	3	$(118, 246)$	$(162, 328)$	$(239, 472)$
-	9	2	$(100, 236)$	$(133, 330)$	$(280, 553)$

The results are given as a pair (a, b) where a refers to the number of iterations for the POMA to reach the first correct classification and b refers to the case where the POMA has fully converged.

In all experiments, the number of states of the POMA is set to 10.

POMAp$\mathcal{X}$: $\mathcal{X}$ refers to the Environment's probability of generating samples within the same class, i.e. POMAp9 means $p = 0.9$.

N: Number of objects to be partitioned.

W/R: Number of objects in every class.

R: Number of classes in the partitioning problem.

The value used for τ is $\tau^* = \frac{1}{W^2}$, and the value used for κ is $\kappa^* = R\left[(\frac{W}{R})^2 - \frac{W}{R}\right]$.

Source: Excerpt from [55]

Algorithm 15 SkeletalEOMA($\{\xi_p\}, A_i, A_l$)

Input:

- The abstract objects $\{O_1, \cdots, O_W\}$; ξ_i and ξ_j are the states associated with A_i and A_j respectively.
- The number of states N per action.

Output:

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

```
1: begin
2:     if ξ_i div N = ξ_j div N then          ▷ The partitioning is rewarded
3:         Call ProcessRewardEOMA({ξ_p}, A_i, A_j)
4:     else                                   ▷ The partitioning is penalized
5:         Call ProcessPenaltyEOMA({ξ_p}, A_i, A_j)
6:     end if
7:     return {ξ_p}
8: end
```

Source: Excerpt from [56]

Self-organization is the ability for a list, for example, 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. The goal of this re-organization, among others, is to minimize the asymptotic cost or access-time of record retrieval. In the design of adaptive lists, it is assumed that the Environment will not request a record absent from the list, and that each record is retrieved at least once [24].

A list data structure is defined as a structure containing a sequence of elements, $R = \{i : 1 \leq i \leq J\}$. The average access-cost of the list is given as:

$$C(\Pi) = \sum_{1 \leq i \leq J} s_i \pi_i \tag{2.3}$$

where,

Table 2.5: Experimental results for the TPEOMA approach done for an ensemble of 100 runs.

W	W/R	R	TPEOMAp9	TPEOMAp8	TPEOMAp7
4	2	2	(2,24)	(2,30)	(3,40)
6	2	3	(4,41)	(4,51)	(5,64)
-	3	2	(6,37)	(8,50)	(13,74)
8	2	4	(7,57)	(7,71)	(8,91)
-	4	2	(14,50)	(25,78)	(41,125)
9	3	3	(19,65)	(21,78)	(29,113)
10	2	5	(8,75)	(10,95)	(14,121)
-	5	2	(26,69)	(41,92)	(76,178)
12	2	6	(12,95)	(15,123)	(18,155)
-	3	4	(30,91)	(37,110)	(52,155)
-	4	3	(34,86)	(47,107)	(66,157)
-	6	2	(43,86)	(62,121)	(111,209)
15	3	5	(48,123)	(61,159)	(81,203)
-	5	3	(51,101)	(71,133)	(105,205)
18	2	9	(20,156)	(28,199)	(36,275)
-	3	6	(66,153)	(85,194)	(126,283)
-	6	3	(63,126)	(95,170)	(136,244)
-	9	2	(77,129)	(148,222)	(268,391)

The results are given as a pair (a, b) where a refers to the number of iterations for the TPEOMA to reach the first correct classification and b refers to the case where the TPEOMA has fully converged.

In all experiments, the number of states of the TPEOMA is set to 10.

$TPEOMAp\mathcal{X}$**:** $\mathcal{X}$ refers to the Environment's probability of generating samples within the same class, i.e. TPEOMAp9 means $p = 0.9$.

N**:** Number of objects to be partitioned.

W/R**:** Number of objects in every class.

R**:** Number of classes in the partitioning problem.

The value used for τ is $\tau^* = \frac{1}{W^2}$, and the value used for κ is $\kappa^* = R\left[(\frac{W}{R})^2 - \frac{W}{R}\right]$.

Source: Excerpt from [56]

- s_i is the unknown probability of the accessing element 'i' - as defined by the probability distribution of the Environment, and
- π_i is the position of element with index i in the list.

The probability distribution of the query accesses from the Environment must be non-uniform. If it was uniform, list re-ordering have no intrinsic worth as the average access-cost would remain unchanged [24].

The following sub-sections will review the performance analysis techniques for list re-organization algorithms, their convergence criteria, as well as a number of deterministic and probabilistic adaptive list algorithms. The simplest and yet most prominent among them are the Move-to-Front (MTF) and the Transposition rule (TR) adaptive schemes, which serve as the focal constituents of the research methodologies in the design of hierarchical adaptive lists in Environments possessing "locality-of-reference", which is also the main concern of this work.

2.4.1 Performance Evaluation

The optimal access-cost for a list can be proven to be when the list sequence is kept in the descending order of their access probabilities. This formation ensures that the more frequently accessed records are placed towards the front of the list. In evaluating the list access cost, one invokes the paid exchange cost model which takes into account the cost incurred in list re-ordering as part of the reported access cost.

On the other hand, one can also consider the standard (or free exchange) cost model [2]. This model is more pervasive because estimating the list access cost is more or less constant with no marginal effect due to the cost of a single reordering operation. The following sub-sections will review two standard cost models that the literature employs in evaluating list access costs - the asymptotic cost and the amortized cost.

Asymptotic Cost Strategies

The expected search cost or asymptotic cost of a list of J elements assumes that the list re-organization strategy converges. Thus, the formal expression for the asymptotic

cost of an adaptive strategy, A, is given as:

$$E[A] = \sum_{1 \le j \le J!} P\{\pi_j\}_A \, C(\pi_j) \tag{2.4}$$

$$= \sum_{1 \le j \le J!} [\, P\{\pi_j\}_A \sum_{1 \le i \le J} s_i \pi_j(i) \,]. \tag{2.5}$$

In Equation (2.5), the expression $P\{\pi_j\}_{\mathcal{A}}$ represents the steady-state (or stationary) probability of choosing the list permutation π_j in the Markov chain that involves the adaptive strategy, A. Further, $C(\pi_j)$ is the ordering cost or the average-access cost of the list permutation π_j, as shown in Equation (2.3). To clarify this, we present below, in Figure 2.10, the Markov chain transition model for a list with three elements $\{a, b, c\}$, and where the list re-organization is achieved using the so-called Move-to-Front (MTF) scheme explained in Section 2.4.4.

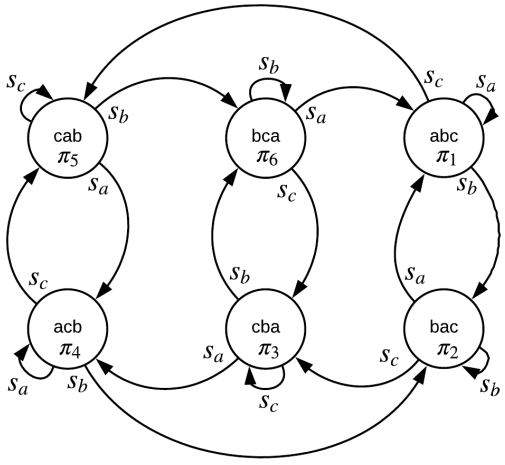


Figure 2.10: The Markov Chain state transition diagram for a list of size three, $\{a, b, c\}$, where the sequences in each state of the chain is a permutation of the elements. The adaptive strategy in this example is the MTF rule.

The diagram in Figure 2.10, has six states, because there are six possible permutations of the elements in the list. Whenever an element in the list is queried or accessed, that element is moved using the re-organization strategy. Since, in this case, we invoke the MTF, the scheme moves the queried element to the head of the list. For example, if the list is in state π_1 and the element c is queried, the list configuration moves to state π_5 with probability s_c. The transition matrix, M, of the Markov chain

produced by the MTF rule is as shown below:

$$M = \begin{array}{c} \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \end{array} \begin{array}{c} \begin{array}{cccccc} \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 & \pi_6 \end{array} \\ \begin{bmatrix} s_a & s_b & 0 & 0 & s_c & 0 \\ s_a & s_b & s_c & 0 & 0 & 0 \\ 0 & 0 & s_c & s_a & 0 & s_b \\ 0 & s_b & 0 & s_a & s_c & 0 \\ 0 & 0 & 0 & s_a & s_c & s_b \\ s_a & 0 & s_c & 0 & 0 & s_b \end{bmatrix} \end{array}$$

It is well known that for an ergodic Markov chain, M, the steady state (stationary) probability vector is computed using the eigenvector $\underline{P}$ when the eigenvalue equals unity, and thus:

$$M^T . \underline{P} = \underline{P} \tag{2.6}$$

where, $\underline{P} = [P\{\pi_1\}\ P\{\pi_2\}\ ...\ P\{\pi_{J!}\}]^T$. This equation can thus be used to compute the stationary probabilities of the various list permutations $\{P\{\pi_j\}\}$, and then be further invoked to obtain the expected cost of the strategy, as in Equation (2.4).

From the above we see that the closed form expression for the asymptotic probability of a permutation can be derived using the theory of Markov chains [13, 36]. Given a list of size J, we have $J!$ list orderings, and a state in the Markov chain represents a particular list permutation $\pi_j, (1 \leq j \leq J!)$, i.e., with a transition matrix M of size $J!$. The re-organization strategy used facilitates the transition from one state to another using the query probabilities, $\{s_i\}$. However, this method quickly becomes computationally infeasible with large list sequences of size J, because it will then need to be evaluated over $J!$ states.

Clearly it is rather difficult to obtain the probabilities of all the possible list permutations, especially when the size of the list increases. The work by [23, 50] employed the strategy of computing the expected value of the list configurations $E[A]$ based on obtaining the probability that element R_j precedes R_i. Let I_{ji} be given as:

$$I_{ji} = \begin{cases} 1 & \text{if } R_j \text{ precedes } R_i \\ 0 & \text{otherwise.} \end{cases}$$

Going by this, R_i's position is given as $1 + \sum_{j \neq i} I_{ji}$. Observe that $E[I_{ji}] = P(R_j \text{ precedes } R_i)$, i.e., the expected value of the probability that R_j precedes R_i. Thus, the asymptotic expected search cost evaluates to:

$$\begin{aligned} E[C] &= \sum_{1 \leq i \leq n} (\ s_i \ (1 + \sum_{j \neq i} P(R_j \text{ precedes } R_i)) \ ) \\ &= \sum_{1 \leq i \leq n} (\ s_i + s_i \sum_{j \neq i} P(R_j \text{ precedes } R_i) \ ) \\ &= 1 + \sum_{1 \leq i \leq n} \sum_{j \neq i} (\ s_i \ P(R_j \text{ precedes } R_i) \ ). \end{aligned} \tag{2.7}$$

From Equation (2.7), given any two competing adaptive list strategies α and β, a winning strategy is chosen by showing that for an entry $\langle R_i, R_j \rangle$, the probability $P(R_j \text{ precedes } R_i)$ is greater whenever $s_j > s_i$.

Given the Markov transition matrix produced from the chain in Figure 2.10, if we only focus on permutations where list entry a precedes b, the equivalent chain now possesses only two states. The first state contains all the list permutations where a precedes b and the second, where b precedes a. This reduced matrix is shown below, and the Markov chain is illustrated in Figure 2.11.

$$M = \begin{array}{c} \\ \pi_1 \\ \pi_2 \end{array} \begin{array}{c} \begin{array}{cc} \pi_1 & \pi_2 \end{array} \\ \begin{bmatrix} 1 - s_b & s_b \\ s_a & 1 - s_a \end{bmatrix} \end{array}$$

The reduced matrix is more tractable to solve using Equation (2.6), and results in the steady-state probability vector $\underline{P} = (\frac{s_a}{(s_a+s_b)}, \frac{s_b}{(s_a+s_b)})^T$. This implies that using the MTF, the probability of the list permutation being in a state where a precedes b is $\frac{s_a}{(s_a+s_b)}$. In any case, the results of [13] observe that this method works especially

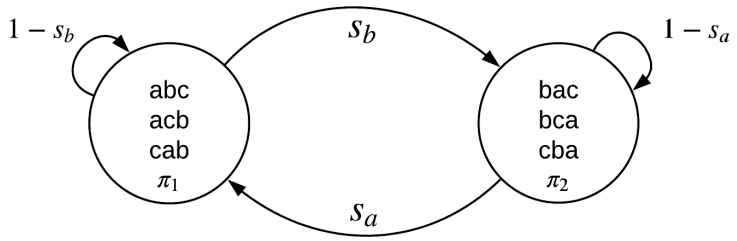


Figure 2.11: The reduced Markov Chain state transition diagram produced by the MTF rule. The first state represents when list configurations a precedes b, and the second when b precedes a.

well with the Move-to-Front (MTF) update scheme but does not scale particularly well for other re-organization algorithms.

Some Markov chains have a convenient property called *time reversibility* [51], where for a given state, the probability of returning to that state in one direction is the same as the probability in the opposite direction. For a Markov chain M, with elements $[m_{ij}]$, the chain is *time reversible* iff:

$$m_{i,i_1}\ m_{i_1,i_2}\ ...\ m_{i_{k-1},i_k} = m_{i,i_k}\ m_{i_k,i_{k-1}}\ ...\ m_{i_1,i},$$

for all states $i, i_1, ..., i_k$. To further illustrate the concept of *time reversibility*, consider a Markov chain transition diagram for a list where each state is a list permutation, and the update rule is the TR rule as shown in Figure 2.12.

In Figure 2.12, the chain can move in a clockwise direction from $\pi_1 \rightarrow \pi_2 \rightarrow \pi_3 \rightarrow \pi_4 \rightarrow \pi_5 \rightarrow \pi_6 \rightarrow \pi_1$ with probability $s_b s_c s_c s_a s_a s_b = s_a^2 s_b^2 s_c^2$. On the converse, going from $\pi_1 \rightarrow \pi_6 \rightarrow \pi_5 \rightarrow \pi_4 \rightarrow \pi_3 \rightarrow \pi_2 \rightarrow \pi_1$, the probability is $s_c s_c s_b s_b s_a s_a = s_a^2 s_b^2 s_c^2$. Hence, we evaluate to the same probability value, even though we move in the opposite direction.

Time reversible Markov chains have the following elegant property given in Equation (2.8) first described in [50] for analyzing the TR rule. It was later used and generalized in [41] for analyzing the Swap-with-Parent (SWP) adaptive scheme. For

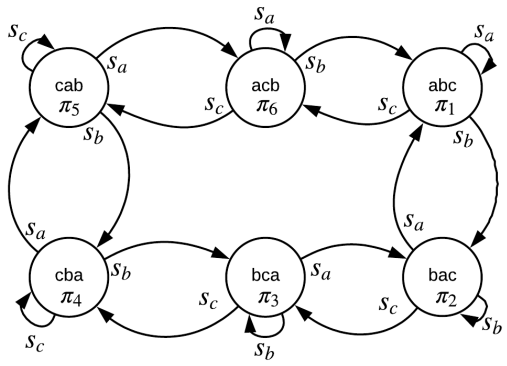


Figure 2.12: The Markov Chain state transition diagram produced by the TR rule.

a time-reversible adaptive list with transition matrix $M = [M_{ij}]$,

$$P\{\pi_i\}m_{i,j} = P\{\pi_j\}m_{j,i} \quad \forall\, i \neq j. \tag{2.8}$$

The expression in Equation (2.8) gives a method for evaluating the transition probabilities of two distinct list permutations and is vital in determining $P(R_j \text{ precedes } R_i)$.

In general, empirical measures are also used for assessing the algorithmic performance of re-organization strategies [3]. This is performed by implementing the corresponding strategy and taking the ensemble mean asymptotic cost. Since the schemes are ergodic, they can be seen to provide an accurate estimation of the asymptotic cost.

Amortized Cost Strategies

The Amortized cost evaluates the performance of list adaptive strategies by taking the mean over all query costs [7, 24, 57]. This evaluation strategy is particularly relevant in evaluating list re-organization in NSEs, where the actions are stochastic as time varies. In [37], the optimality of a scheme is determined by minimizing:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{T} E[\beta(n)],$$

where, $E[\beta(n)]$ is the expected cost at time instance n, which is the description of the amortized cost.

2.4.2 Convergence

A convergence criterion termed "*overwork*" has also been proposed for self-organizing list schemes [8]. *Overwork* is the region between the cost curve and its asymptote. A steeper arch will indicate a faster rate of convergence for the re-organization scheme, and a smaller *overwork* area. The literature recommends that we evaluate the space under the cost curve when comparing competing schemes [8].

It should be noted that a closed-form expression for the overwork is computationally infeasible except for a restricted set of adaptive schemes [3]. For this reason one typically considers alternative empirical measures. It is customary, in the literature, that we execute the adaptive scheme for a large number of iterations, and then choose the final time-average cost as the asymptotic cost. However, the rate of convergence is not a critical criterion in designing adaptive schemes.

2.4.3 Deterministic Adaptive Approaches

This sub-section surveys a few of the already-existing classes of adaptive list re-organization schemes reported in the literature. A relatively simple approach to re-organize a list that uses additional memory, is the *Frequency Count* (FC) scheme. The FC maintains an accumulator for recording the access frequencies of the list elements. The resulting list is re-arranged according to the descending order of the counters. This relatively simple scheme yields rather impressive results with regard to its asymptotic cost being close to optimal, and its amortized cost being about two times the optimal cost [7]. Notwithstanding this, the FC scheme has some obvious drawbacks, the first being the memory cost of $\mathcal{O}(n)$, and that it scales poorly for large lists [30]. Further, in environments exhibiting locality of reference, the FC scheme yields an unacceptable performance [24, 41].

The following two sub-sections review the prominent MTF and the TR schemes. Their importance to this research warrants their review in a sub-section on its own.

2.4.4 Move-To-Front (MTF)

To re-iterate the rule, the MTF update heuristic moves the queried element to the front of the list except when it is the first record, in which case, by default it is already at the front.

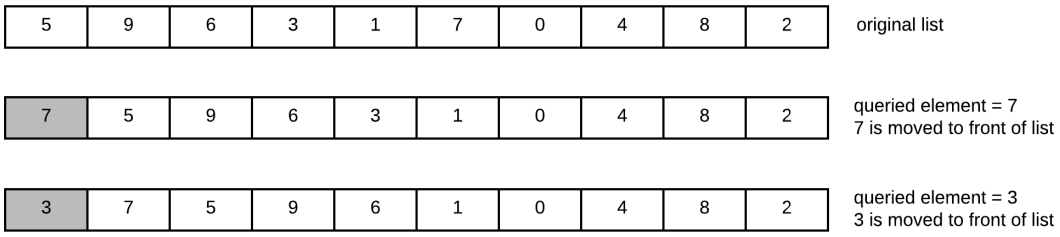


Figure 2.13: A diagrammatic description of the Move-To-Front (MTF) rule.

The search cost for the MTF is analyzed in [8, 30, 36, 50], with a closed form solution to the probability $P(R_j \text{ precedes } R_i)$ derived using the theory explained above. A simpler derivation of Equation (2.7) is presented in [23, 50], which we now briefly review. If R_j is queried before R_i with an elapsed time of k units, we know that there have been $k-1$ requests for other records discounting R_i and R_j. By summing the probability over all k, we get:

$$\begin{aligned} P(R_j \text{ precedes } R_i) &= s_j * \sum_{1 \le k \le \infty} (1 - s_i - s_j)^{(k-1)} \\ &= \frac{s_j}{s_i + s_j}. \end{aligned}$$

By applying, Equation (2.7) to the above equation, we obtain the average cost of the MTF, C_{MTF}, as:

$$
\begin{aligned}
C_{MTF} &= 1 + \sum_{1 \le i \le n} \sum_{j \neq i} (s_i \; P\{R_j \text{ precedes } R_i\}) \\
&= 1 + \sum_{1 \le i < j \le n} \sum (s_i \; P\{R_j \text{ precedes } R_i\} + s_j \; P\{R_i \text{ precedes } R_j\}) \\
&= 1 + \sum_{1 \le i < j \le n} \sum (\frac{s_i s_j}{s_i + s_j} + \frac{s_j s_i}{s_i + s_j}) \\
&= 1 + 2 \sum_{1 \le i < j \le n} \sum \frac{s_i s_j}{s_i + s_j}.
\end{aligned}
$$

Note that the average cost of the optimal ordering is given as $C_{OPT} = \sum_{1 \le i \le n} (s_i * i)$. The average cost of the MTF update rule is within two-times the average cost of the optimal ordering by taking the ratio C_{MTF} to C_{OPT} [50]. However, a tighter bound was derived in [11], which showed that $\frac{C_{MTF}}{C_{OPT}} \le \frac{\pi}{2} \approx 1.5708$ for any query distribution. As opposed to this, the amortized cost is proven to be at most two-times the average cost of the optimal permutation if the list is initially arranged in order of first access; e.g., if the initial list ordering has four elements $l = \{3, 1, 4, 2\}$, and the query access comes in the same order as the list ordering. [7].

2.4.5 Transposition Rule (TR)

In the Transposition adaptive scheme, TR, a queried record is moved one position towards the front of the list. If the element R is in index i, it is moved to index $i - 1$ for every query request, except if it is at the front of the list.

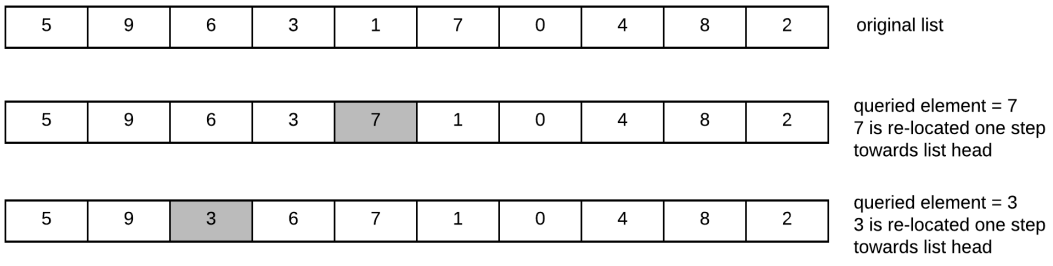


Figure 2.14: A diagrammatic representation of the Transposition Rule (TR).

The TR rule involves a *time reversible* Markov chain, thus simplifying its cost analysis. Indeed, the scheme's stationary probabilities obey:

$$\frac{P\{\pi_a = < R_{i_1} R_{i_2} \ldots R_{i_j} R_{i_{j+1}} \ldots R_{i_n} >\}}{P\{\pi_b = < R_{i_1} R_{i_2} \ldots R_{i_{j+1}} R_{i_j} \ldots R_{i_n} >\}} = \frac{s_{i_j}}{s_{i_{j+1}}}. \tag{2.9}$$

for $1 \leq j < n$, if $s_k \neq 0$ for $1 \leq k \leq n$.

Equation (2.9) shows that the proportion of the probability estimate of a list permutation prior to and after the transposition of a pair of elements is equivalent to the proportion of their respective access probabilities. Given a set of list permutations, π_a and π_b, we can derive the ratio of their probabilities by the following relation:

$$\frac{P\{\pi_a\}}{P\{\pi_b\}} = \frac{P\{\pi_a\}}{P\{\pi_{i_1}\}} \times \frac{P\{\pi_{i_1}\}}{P\{\pi_{i_2}\}} \times \ldots \times \frac{P\{\pi_{i_{k-1}}\}}{P\{\pi_{i_k}\}} \times \frac{P\{\pi_{i_k}\}}{P\{\pi_b\}}$$

for some $1 \leq k \leq n! - 2$. From Equation (2.9), this can be rewritten as:

$$\frac{P\{\pi_a\}}{P\{\pi_b\}} = \prod_{u,v} \left(\frac{s_u}{s_v}\right),$$

for some $1 \leq u, v \leq J!$. Let $\delta(\pi_\alpha, \pi_\beta)$ represent the movement of R_i from π_α to π_β during the transformation. The reader must observe that the *quantity* $\delta(\pi_\alpha, \pi_\beta)$ is positive when the relevant list element is moved away from the list head, and negative for the converse scenario. In the above equation, given a transposition of R_i, s_i shows up in the numerator when $\delta(\pi_\alpha, \pi_\beta)$ is positive and in the denominator when negative. Consequently, s_i can be re-written in terms of $s_i^{\delta_i(\pi_a, \pi_b)}$. We now have:

$$\frac{P\{\pi_a\}}{P\{\pi_b\}} = \prod_{1 \leq i \leq n} s_i^{\delta_i(\pi_a, \pi_b)}.$$

By substituting Equation (2.5) into the above equation which yields the probability of any list permutation, π, w.r.t. the original arrangement, π_0, we can derive the expected cost for the TR adaptive scheme[4] as, A_{TR} [50]:

[4]The reader is referred to [13] for the details of the analysis.

$$A_{TR} = P\{\pi_0\} \sum_{\pi} \left(\left(\prod_{1 \le i \le n} s_i^{\delta_i(\pi_0,\pi)} \right) \sum_{1 \le j \le n} s_j \pi(j) \right)$$

where

$$P\{\pi_0\} = \left(\sum_{\pi} \prod_{1 \le i \le n} s_i^{\delta_i(\pi_0,\pi)} \right)^{-1}.$$

When compared to the MTF, TR has a lower asymptotic cost except for lists with uniform probabilities or lists with only two elements [50]. Moreover, it was conjectured [50] that TR minimized the mean query cost for every distribution. However, the authors of [4] found a counter-example to disprove this conjecture.

An empirical analysis of TR responding to query accesses from different probability distributions shows that TR outperforms MTF for the Zipf's distribution [50], and the results of [13] showed that the asymptotic cost of TR outperforms MTF for the Lotka, exponential, linear, and 80-20 probability distributions. The results are presented in Table 2.6. Indeed, one can see that:

$$A_{TR} \le A_{MTF} \le \frac{\pi}{2} * A_{OPT}.$$

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	26.439	31.703	4.484	2.7722	41.735
TR	19.717	24.237	3.419	2.0357	34.515

Table 2.6: Experimental results for the asymptotic cost of the MTF and TR rules for lists of size 100 [3, 13].

TR is, however, slower in converging towards the optimal ordering. By employing the *overwork* technique, the authors of [8] showed that for a given distribution where $s_1 = 0$ and $s_i = 1/(n-1), 2 \le i \le n$, the *overwork* for the TR rule is $(n^2-1)/6$ and for MTF rule is $(n-1)/2$ for the MTF rule. Hence, the difference between the TR and MTF adaptive schemes are of an *order of magnitude*.

Table 2.7, shows the empirical slow rate of the TR against the MTF [3, 13].

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	9,000	30,000	3,000	5,500	1000
TR	170,000	160,000	250,000	270,000	130,000

Table 2.7: Experimental results for the convergence of MTF and TR rules for lists of size 100 [3, 13]

2.4.6 Composite MTF & TR Strategies

The MTF and TR rules have obvious advantages and drawbacks. The MTF has a faster adaptive rate and quickly converges early-on in the algorithm's execution. As opposed to this, the TR is guaranteed to converge more accurately towards the optimal list ordering based on the probability distribution of query accesses. The MTF is, however, superior in Environments with dependent query accesses (i.e., with locality of reference) [10], as we shall see presently.

The first of such a composite scheme was proposed in [8], where the scheme begins the list re-organization by using the MTF rule for a quick convergence, and thereafter, at a particular instance, say, k, switches to the TR to ensure a superior convergence. While this is effective, the choice of the parameter k, to make the switch, becomes a context-based engineering judgment.

Another hybrid solution is the Move-ahead-k, MHD(k) [50], proposed as a trade-off between the MTF and TR schemes. MHD(k) moves a queried element k positions to the front of the list. When the element is within the first k positions, the element is moved to the front of the list. However, the choice of k is again a context-based engineering decision, primarily determined by the size of the list [58]. While a formal analysis of this scheme is difficult, its performance is meant to be better than MTF and faster than TR[5].

Lastly, among the hybrid solutions, are the POS(k) and the SWITCH(k) rules [59]. In the POS(k) scheme, when an element is queried, it is relocated to the k^{th} position of the list provided $j > k$; otherwise, it is transposed if the element is in positions 2 to k. The SWITCH(k) scheme is similar to the POS(k), but employs the MTF when the element is in positions 2 to k. The reader will readily observe that POS(1)

[5]Note that MHD(1) becomes the TR rule.

becomes the MTF and POS$(n-1)$ becomes the TR.

2.4.7 Other Deterministic Adaptive Strategies

This sub-section will review other list re-organization strategies from literature. The first of such is the Swap-with-Parent (SWP), and the Move-to-Parent (MTP) adaptive schemes [13, 41]. These schemes take into consideration the size of the list in the re-organization process, which is reported to hold sway in the convergence power of an adaptive scheme [58]. The SWP takes a conceptual view of the list as a binary tree. This tactic allows the SWP to address the list without an ordering constraint. The SWP swaps a queried element with its parent in the tree. The parent record of an element, i, is the record half the distance between the front of the list and i. A frequently accessed record can be brought to the head of the list with as little as $\lfloor \lg n \rfloor$ accesses.

An advantage of SWP is that it improves the convergence speed against the TR, and also avoids the instability of the MTF. The SWP has been empirically shown to have superior asymptotic cost to the MTF. However, a drawback associated with the SWP is that for an element with zero-probability, the element is never swapped if it is not a parent of another element. Thereby, this results in list elements (sometimes almost up to half of the list size) with higher access probabilities placed behind this element.

The MTP scheme mitigates this deficiency by moving the queried element to its parent's position, while the parents and all other preceding elements are moved back one step. This strategy works like an automated MHD(k). Table 2.8 and Table 2.9 present some empirical results for the SWP and MTP schemes [3, 13]. The former compares their asymptotic costs, and the latter their convergence rates.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
SWP	24.276	29.245	4.235	2.7719	41.379
MTP	22.118	27.008	3.771	2.3235	38.660

Table 2.8: Experimental results for the asymptotic cost of the SWP and MTP rules for lists of size 100 [3, 13].

Scheme	Zipf	80-20	Lotka	Exponential	Linear
SWP	25,000	20,000	30,000	20,000	2,500
MTP	20,000	30,000	20,000	60,000	20,000

Table 2.9: Experimental results for the convergence rates of the SWP and MTP rules for lists of size 100 [3, 13]

Batched Strategies

The next set of procedures are called "Batched" algorithms in the literature because they are used in combination with adaptive schemes to determine if a re-organization should occur. One of these is the move-every-kth-access rule [36]. This batched strategy attaches a counter to every record in the list which is accumulated per query. When the value of each counter reaches k, the associated element is moved to the front of the list and the counter is set to zero (i.e., assuming the MTF is the affiliated adaptive scheme in operation). This conjunction results in a scheme with a lower cost than the standalone MTF barring queries from a uniform distribution [8]. A further modification resets all counters to zero when k is reached. In this modification, the asymptotic cost is close to the optimal cost as $k \to \infty$ [8].

Another batched technique is the k-in-a-row scheme [29]. This method re-organizes the list after an element has been queried k consecutive times. It maintains two counters for the last element queried and the counter k. Hence, the memory requirement is $\mathcal{O}(1)$. The asymptotic cost is close to being optimal as $k \to \infty$. It is further proven that $(k+1)$-in-a-row is superior to k-in-a-row [22].

The last of such batched procedures that this research will review is the k-in-a-batch technique; which groups queried elements into batches of size k. The adaptive scheme is executed when all the queries within a batch are identical. This batch strategy has the same memory requirement as the k-in-a-row. The k-in-a-batch associated with the MTF or TR is proven to be superior to the k-in-a-row w.r.t. the asymptotic cost.

TS(0)

TS(0) with (TS for *time-stamp*) is the deterministic variant of the TS(p) class of randomized algorithms [1, 2]. The TS(0) approach is to insert the queried element, x in front of the first list record queried at most once since x was last queried. If no such element exists, then we leave x "as is". The TS(0) has been shown to be superior to the MTF w.r.t the asymptotic cost [2]. It, however, requires a memory overhead of $\mathcal{O}(n)$, which is an undesirable property.

MRI(m) & **PRI(**m**)**

The MRI(m) and PRI(m) represent two classes of adaptive schemes called Move-to-Recent and Pass-Recent-Item [5]. MRI(m) relocates the queried element x to a position ahead of the last element, y in the list that is in front of x which was requested at least $m + 1$ times since x was last requested. If this is the initial request for x, or in the absence of the move condition, x is moved to the head of the list. MRI(m) suffers the same $\mathcal{O}(n)$ memory drawback observed in TS(0).

PRI(m), on the other hand, moves the queried element x to a position ahead of the first element y in the list that was requested at most m times since x was last requested. If this is the initial request for x, x is moved to the head of the list. However, in the absence of the move condition, x stays at its same position. Apart from the initial query, MRI(1) and PRI(1) resolve to TS(0).

SBR(α**) & SBD(**k**)**

The SBR(α) and SBD(k) represent another two similar classes of adaptive schemes named the Sort-by-Rank and Sort-by-Delay algorithms [52]. Central to these algorithms is the quantity $w_k(x, t)$ which is defined as the k^{th} last access to record x requested at time t. A rank function $s_k(x, t) = t - w_k(x, t)$ is measured as the time that has transpired since the last k^{th} access on element x at time t. Hence, for $\alpha \geq 0$, the rank function is set as a convex combination of $s_1(x, t)$ and $s_2(x, t)$:

$$r_t(x, \alpha) = (1 - \alpha) \cdot s_1(x, t) + \alpha \cdot s_2(x, t).$$

In other words, the rank function is a convex combination of the time following the last access and the access just before the last, with α being a weight differentiator.

Given a query request for element x at time t, SBR(α) places x at the position after the last element y that is in front of x with $r_t(y, \alpha) < r_t(x, \alpha)$. If there is no element in the preceding condition or if x is queried first, it moves x to the head of the list. SBD(k) on the other hand places x at the position after the last element y that is in front of x so that $s_k(y, t) < s_k(x, t)$. If there is no element in the preceding condition or if x is queried first, it moves x to the head of the list. SBD(k) has been proven to be asymptotically optimal as $k \to \infty$.

Move-To-Rear(MTR)

An ergodic Markov chain cannot be used to describe all adaptive list schemes. The literature also reports certain *absorbing* schemes [44, 45, 48]. An example of a scheme represented by an absorbing Markov chain is the non-intuitive Move-to-Rear (MTR) rule.

The MTR class of algorithms relocates queried elements to the *end* of the list as opposed to the head. This action has an ensuing terminal property of placing the list in its optimal ordering asymptotically, and in the deterministic version, each element is moved just once in the entire permutation, thereby drastically curtailing re-organization time.

The first of the MTR family reviewed here is the Deterministic linear space MTR scheme [45]. This scheme is fundamentally a move-every-kth query rule; although an element is relocated *only once* to the end of the list after k query requests. For every record i, the scheme keeps a counter x_i and a Boolean m_i to maintain the frequency of queries and to check when an element is already relocated. If $s_i > s_j$ given that either element is accessed, the asymptotic probability that an element A_i outranks A_j, can be made to be as near to unity as desired.

We now consider the Deterministic constant space MTR scheme [45]. In this scheme, queried elements are relocated *only once* to the end of the list when accessed k consecutive times. A Boolean variable is not necessary for this scheme; rather, one maintains only the counter that tracks the last queried record. It thus, merely,

requires $\mathcal{O}(1)$ additional space. The Boolean variable is conveniently eliminated by keeping track of the position of the very first element that was moved to the end of the list, called *FrontNew*. Any element that precedes this element (i.e., *FrontNew*) has not been permutated. This scheme has been proven to be expedient and conjectured to be optimal.

Another scheme in this class, called the Optimal deterministic constant space MTR [48] enhanced the deterministic constant space move-to-rear scheme to be optimal asymptotically. Just like the deterministic constant MTR algorithm, query elements are relocated to the end of the list when queried k consecutive times, with the exception that the optimal deterministic constant MTR scheme can locate elements in-between consecutive requests that have already been migrated. The query counter is reset when a query for an element that has not already been permutated (i.e., moved to the end of the list) is made.

2.4.8 Probabilistic Approaches

In this section, we will review several probabilistic re-organization strategies reported in the literature. One of such is the probabilistic MTF and MTR schemes. A probabilistic version of the MTF scheme called the *constant space probabilistic MTF* relocates records to the front of the list when queried, but it does so with a diminishing probability distribution [44]. Given an initial condition $f(0) = a$, where $0 < a < 1$, for every time step, the expression below updates the probability:

$$f(n+1) = af(n),$$

where $f(n)$ is a function that represents the probability that a queried element is moved to the head of the list at time n. A single memory location is needed to store the function $f(n)$. Since, $f(n)$ can also be evaluated as $f(n) = a^n$, what is stored simply becomes the index n, of the time that transpired since the execution of the adaptive scheme. The *constant space probabilistic MTF* scheme is shown to be expedient; however, the deterministic version boasts a superior performance.

The *linear space probabilistic MTR* scheme non-intuitively relocates the queried

element A_i to the rear of the list with probability q_i, which diminishes every time A_i is requested [44]. The formal expression is as shown below:

$$q_i(n+1) = \begin{cases} aq_i(n) & \text{if } A_i \text{ is accessed,} \\ q_i(n) & \text{otherwise,} \end{cases}$$

where $0 < a < 1$. In this scheme, the list operations decline as the probability of access (that moves a requested element to the back of the list) tend to zero (asymptotically). Hence, the reorganization time is cut short. It has been proven that for any i and j,

$$E[q_j(n)] < E[q_i(n)] \text{ if and only if } s_j > s_i.$$

Also, if $s_j > s_i$, the probability that $q_j(n) < q_i(n)$ can be made as close to unity as desired as well as the probability of the list converging to the optimal arrangement.

In the interest of completeness, we will also merely mention a few randomized probabilistic schemes. One of these is the $SPLIT$ algorithm [27]. In this scheme, every element x keeps a link to a certain element $x.split$ and moves x to the front of the list with probability 0.5, else it places the element in front of the last known element at the front of the list. Other sets of algorithm classes that employ both deterministic and probabilistic strategies are the JUMP [25], MTF2, Randomized MTF (RMTF), and the Randomized move ahead (RMHD) schemes [5] omitted here as they are not relevant to our research.

2.4.9 Concluding Remarks

In this chapter, we surveyed the relevant field of Learning Automata (LA) that sets the framework for the reinforcement schemes used in this work. In particular, we examined the concept of NSEs for learning in contexts that change with time of which the MSE and PSE environments are of prime interest.

Further, this chapter discussed strategies for addressing the object partitioning problem using LA-based solutions. This research adopts the OMA-family of algorithms in designing a hierarchical adaptive scheme with "sub"-structures due to its improved accuracy and convergence time, although it is constrained to the equal partitioning problem. The vanilla OMA algorithm has a better accuracy than the SMA,

and also a better time and space complexity over the ML_{RP}. Considering accuracy, it is our belief in this work that the state-of-the-art OMA methods, i.e., the PEOMA and TPEOMA will yield even better accuracy estimates.

Finally, we briefly surveyed the existing re-organization strategies for designing adaptive lists including the deterministic approaches and probabilistic approaches. We examined their performance evaluation and convergence criteria. Of greater interest to this work are the Move-To-Front (MTF) and the Transposition Rule (TR) schemes for two reasons; the first being that they have been shown to empirically out-perform the other schemes [5], and the time and space complexities involved in implementing other surveyed schemes render most of them impractical for real-world settings.

For Environments with locality-of-reference, the MTF has been shown to be superior to other schemes such as FC, TR, MRI(0) and TS(0) [5]. However, the MTF still performs better in sequential environments [10]. It is for these sort of Environments with dependent query accesses that this present work seeks to provide novel solutions. This thesis combines the MTF and TR rule to take advantage of the quick updates of the MTF rule and the asymptotically stable convergence of the TR rule in designing our improved hierarchical adaptive strategies.

3

Adaptive Data "Sub"-Structures

3.1 Introduction

The survey in Chapter 2 provided the relevant background to this thesis. In that chapter, with regard to adaptive list re-organization, we introduced the MTF and TR schemes and argued that the MTF is superior to other adaptive schemes in Environments with locality of reference or with dependent query accesses due to its rapid convergence rates. However, TR which is otherwise slower, is more stable and guaranteed to converge more accurately asymptotically.

As stated earlier, the central theme of this work is to design data structures that adapt as they receive queries from an NSE. Hence, this chapter discusses two dependent query generators for modelling an Environment with dependent queries, and addresses the rationale behind the hierarchical design for adaptive data structure modelled as data "sub"-structures.

The chapter concludes by discussing the results reported in [3] that were replicated and verified in this work. We further address the need to improve the hierarchical setup to enable elements to move between the various "sub"-structures by clustering elements that should be in the same data "sub"-structure using a reinforcement scheme.

3.2 Locality of Reference

In NSEs, the penalty probabilities for each action vary with time. In the context of an adaptive data structures, 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]. In other words, there is a considerably small number of distinct or unrelated queries within a segment of the access sequence.

To initiate the design of adaptive data structures in NSEs, we introduce and examine two dependent query generators for simulating an Environment producing queries with dependent accesses. They are the Markovian and Periodic query generators. Given a set of n distinct elements, if we split it 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. The elements within a sub-context k_i exhibit Locality of Reference. This implies that if an element from set k_i is queried at time t, there exists a high likelihood that the next queried element at time $t + 1$ will also arrive from the same set k_i. In other words, the Environment itself can be seen 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.

3.2.1 Markovian Switching Environment (MSE)

Consider, for example an Environment with 128 distinct records, that are divided into $k = 4$ subsets, with 32 contiguous elements in each subset. In such a case, the set of

states referred to above, $\{Q_1, Q_2, Q_3, Q_4\}$, could be $Q_1 = \{1 \dots 32\}$, $Q_2 = \{33 \dots 64\}$, $Q_3 = \{65 \dots 96\}$, and $Q_4 = \{97 \dots 128\}$. The Markovian Switching Environment (MSE) models each subset as the states of the Environment, which are, in turn, the states of a Markov chain. If the probability of the Environment choosing a record from the current subset is 0.9 and the probability of switching to another subset is equally divided among the other three subsets, the transition matrix for the states of the Environment (represented by the Markov chain) is shown below:

$$\begin{array}{c} \\ Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{array} \begin{array}{c} \begin{array}{cccc} Q_1 & Q_2 & Q_3 & Q_4 \end{array} \\ \begin{bmatrix} 0.9 & \frac{0.1}{3} & \frac{0.1}{3} & \frac{0.1}{3} \\ \frac{0.1}{3} & 0.9 & \frac{0.1}{3} & \frac{0.1}{3} \\ \frac{0.1}{3} & \frac{0.1}{3} & 0.9 & \frac{0.1}{3} \\ \frac{0.1}{3} & \frac{0.1}{3} & \frac{0.1}{3} & 0.9 \end{bmatrix} \end{array}.$$

In a more general case, after a query generated from state Q_1, the Environment remains in that state with probability α and moves to a different state with probability $\frac{1-\alpha}{k-1}$. The Markov chain can then be analyzed to describe the behaviour of the Markovian query generator given the fixed probability distribution of state transitions and the parameter α, as shown in the matrix below [5]. Further, the work of [5] showed that $\frac{1}{1-\alpha}$ is the expected number of queries received from the current local set before it switches to another local set.

$$\begin{array}{c} \\ Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{array} \begin{array}{c} \begin{array}{cccc} Q_1 & Q_2 & Q_3 & Q_4 \end{array} \\ \begin{bmatrix} \alpha & \frac{1-\alpha}{k-1} & \cdots & \frac{1-\alpha}{k-1} \\ \frac{1-\alpha}{k-1} & \alpha & \cdots & \frac{1-\alpha}{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1-\alpha}{k-1} & \frac{1-\alpha}{k-1} & \cdots & \alpha \end{bmatrix} \end{array}$$

3.2.2 Periodic Switching Environment (PSE)

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, and the other is when the data structure is unaware of the state change. Empirically, understandably, the performance of the adaptive scheme is better when the data structure is aware of the Environment's state change.

Although the PSE is, in and of itself, a single phenomenon, it appears naturally in real-world settings. Consider, for example, the periodic nature of the volume of customers visiting a super-store during the day. This volume, typically, drops early in the morning and late in the night and then peaks around mid-morning/ late-afternoon. Again, consider the resource allocation of internet bandwidths by an ISP, where, the bandwidth allocation is higher from around midnight to early morning due to the reduced frequency of users on the network, whereas from early-morning to late-evening the resource allocation to the individual users is lower.

3.3 Generating Query Distributions

The Environment generates queries according to a probability distribution. In recording the behaviour of the hierarchical list schemes proposed in this work, we considered five different types of query distributions, namely, the Zipf, Eighty-Twenty, Lokta, Exponential and Linear distributions. For a given list of size J, divided into k sub-lists, with each sub-list 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. Notice that in this way, the total probability mass for the query accesses in each group is the same, and the distribution within each group has the respective distribution. The access probability distribution for these query generators are described below.

1. **The Zipf distribution**: The access probabilities for the Zipf query generator is given as:
$$s_i = \frac{1}{iH_m}, \quad \text{for} \quad 1 \leq i \leq m,$$
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:
$$s_i = \frac{1}{i^{(1-\theta)} H_m^{(1-\theta)}}, \quad \text{for} \quad 1 \leq i \leq m \ \text{ and } \ \theta = \frac{\log 0.80}{\log 0.20} \approx 0.1386,$$
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:
$$s_i = \frac{1}{i^2 H_m^2}, \quad \text{for} \quad 1 \leq i \leq m,$$
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:
$$s_i = \frac{1}{2^i K}, \quad \text{for} \quad 1 \leq i \leq m,$$
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:
$$s_i = K(m - i + 1), \quad \text{for} \quad 1 \leq i \leq m,$$
where K is determined as the constant which normalizes the $\{s_i\}$ to be a valid distribution.

A rationale for conducting the simulations with these query distributions is that, for the most part, they result in "L-shaped" graphs. This is true in particular, for the Exponential and Lotka distribution, and to an extent for the Zipf distributions. Such "L-shaped" distributions assign high probabilities to a small number of the sublist elements. By working in this manner, we can compare our hierarchical variants against the MTF and TR schemes, which were the *de facto* schemes for adaptive lists in NSEs.

3.4 The Rationale for using Data "sub"-structures

This section explains the idea behind formulating the design of adaptive data structures as a hierarchy of data "sub"-structures. We first explore some of the approaches to sublist manipulation and then present the case for a potential solution, and proceed to submit its initial apparent drawbacks. These drawbacks are later mitigated by employing a reinforcement scheme to capture the dependencies between the records.

One approach for framing list organization as a merger of other data structures is to resort to the previously-discussed Swap-with-Parent and Move-to-Parent reorganization strategies. In those cases, the list was formulated as an unordered heap or a tree, and then reorganized based on the parent of the element in the conceptual tree. Actually, this concept arose from the field of self-organizing binary search trees that re-arranges queried elements by rotating them upwards to their parents' node under certain conditions [13].

Another approach is the doubly-exponential tree data structure by Iacono [26] which is built on the concept of *Working Sets* prevalent among algorithms operating in Environments with dependent query accesses, where the algorithm is rather concerned with the data sub-structure (i.e., the Working Set). Iacono's data structure considers a list as a sequence of queues and trees of increasing-sizes.

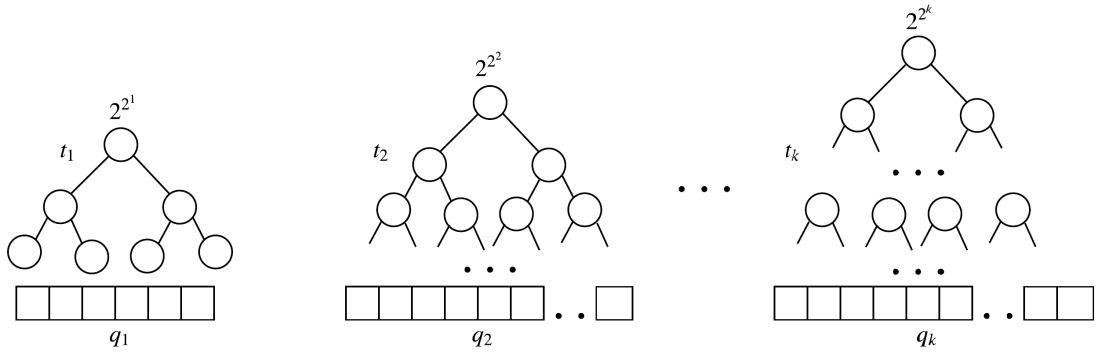


Figure 3.1: Iacono's data structure expressed as a sequence of doubly-exponential queues and the corresponding balanced search trees. The queues arranged in sequence constitute the original list.

The doubly-exponential tree data structure keeps a sequence of balanced search

trees $t_1, ..., t_k$ and a sequence of queues $q_1, ..., q_k$. When linked together, the queues can be viewed as the original list. The work by [26] showed that for any $1 \leq i \leq k$, the size of q_i equals the size of t_i, and the contents of q_i and t_i are identical.

The sequence of trees is doubly-exponential, that implies that the size of $t_i = 2^{(2^i)}$. Iacono's data structure is a unique example of framing self-organizing lists as a series of queues with increasing sizes.

3.4.1 Designing the Hierarchical Schemes

The concept of a hierarchical data "sub"-structure involves dividing a list of size J into k sub-lists. The re-organization strategy is then hierarchically applied to the list by first considering the elements within the sub-list (also called the sub-context) and then operating over the sublists (or sub-contexts) themselves. This hierarchical ordering is one of the fundamental concepts that this thesis builds upon. This research improves the adaptive SLLs on SLLs data structure to further minimize the access query costs.

The re-organization strategies involved are the MTF and TR rules. When used in a hierarchical scheme, this yields 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 list context. Again, the fundamental idea of combining the MTF and TR schemes in this hierarchical formulation is principally to take advantage of the fast convergence properties of the MTF rule and the more accurate asymptotic convergence of the TR rule. More details about these schemes are found in the following sections.

In a NSE with Locality of Reference, given a query access for m_i from Q_a, the probability that the next query access, m_j will come from the same local context, Q_a is high. Hence, it is useful to take advantage of this dependency relationship by moving the entire sublist of elements of the sub-context *en masse* towards the head of the list to cut-down the access time cost when an element within the sub-context Q_a is requested. The hierarchical schemes mentioned above are preferred in Environments

characterized by such a Locality of Reference. Observe that the stand-alone MTF will require at least $\frac{J}{k}$ distinct requests to promote the entire sub-context to the head of the list.

Further, if a record m_u is accessed that is not in the re-organized sub-context, the hierarchical schemes will promote *en masse* all records that are part of m_u's sub-context towards the head of the list thereby reducing the subsequent access costs. As opposed to this in the MTF and TR schemes, for example, the entire context is promoted towards the list head one record at a time.

3.4.2 Challenges of the Hierarchical Schemes

The hierarchical schemes on their own, however, have a major limitation, and as we will soon see from the experiments, they perform worse than stand-alone schemes such as the MTF and TF in NSEs. The drawback is due to the fact that the hierarchical schemes make an assumption 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 in an arbitrary manner. 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. This will be discussed later in this thesis.

3.5 Hierarchical List Reorganization Strategies

This section goes over the hierarchical sub-list re-organization schemes, their algorithmic implementations and the experimental results obtained in NSEs. We also compare their performance with one another and with the stand-alone MTF and TR adaptive schemes. For each hierarchical adaptive scheme discussed, a list of size J is divided into k sublists, where each sublist contains $\frac{J}{k}$ elements.

3.5.1 MTF-MTF

In the MTF-MTF hierarchical adaptive scheme, the accessed element within the sub-context is first promoted to the head of the sub-list, and the entire sub-list is thereafter promoted *en masse* to the head of the main list. This strategy is further illustrated in Figure 3.2.

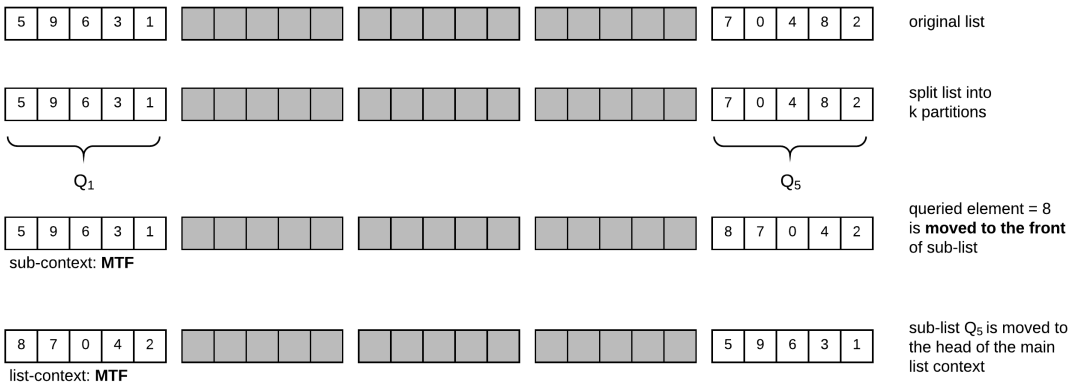


Figure 3.2: A diagrammatic description of the MTF-MTF Hierarchical scheme

The algorithmic implementation for the MTF-MTF is shown in Algorithm 16.

Algorithm 16 The MTF-MTF Algorithm

Input:

- The list-on-list data structure with equi-partitioned elements, A.
- The stream of queries from the Environment, Q.

Output:

- The updated list using the MTF-MTF rule.

1: **begin**
2: **repeat**
3: Receive queried element q_i
4: Get sub-list partition, k containing the query q_i
5: Move element q_i to the head of sub-list, k.
6: Move sub-list k to the head of the list, A.
7: **until forever**
8: **end**

3.5.2 MTF-TR

In the MTF-TR hierarchical adaptive scheme, the accessed element within the sub-context is first promoted to the head of the sub-list, and then the entire sub-list is promoted *en masse* one step towards the head of the main list. This strategy is further illustrated in Figure 3.3.

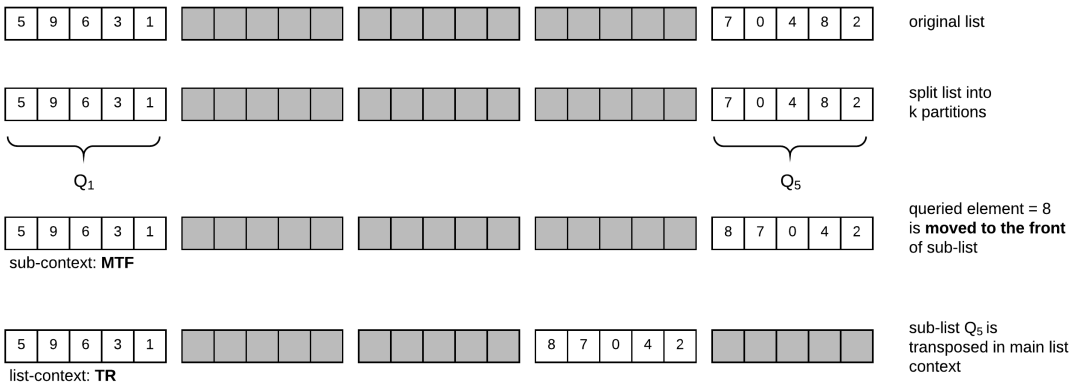


Figure 3.3: A diagrammatic description of the MTF-TR Hierarchical scheme

The algorithmic implementation for the MTF-TR is shown in Algorithm 17.

Algorithm 17 The MTF-TR Algorithm

Input:

- The list-on-list data structure with equi-partitioned elements, A.
- The stream of queries from the Environment, Q.

Output:

- The updated list using the MTF-TR rule.

1: **begin**
2: **repeat**
3: Receive queried element q_i
4: Get sub-list partition, k containing the query q_i
5: Move element q_i to the head of sub-list, k.
6: **if** index(k) ! $= 0$ **then** ▷ If k is not the list head
7: Swap sub-list k with its preceding sub-list, $k-1$.
8: **end if**
9: **until forever**
10: **end**

3.5.3 TR-MTF

In the TR-MTF hierarchical adaptive scheme, the accessed element within the sub-context is first moved one step towards the head of the sub-list, and then the entire sub-list is promoted *en masse* to the head of the main list. This scheme is further illustrated in Figure 3.4.

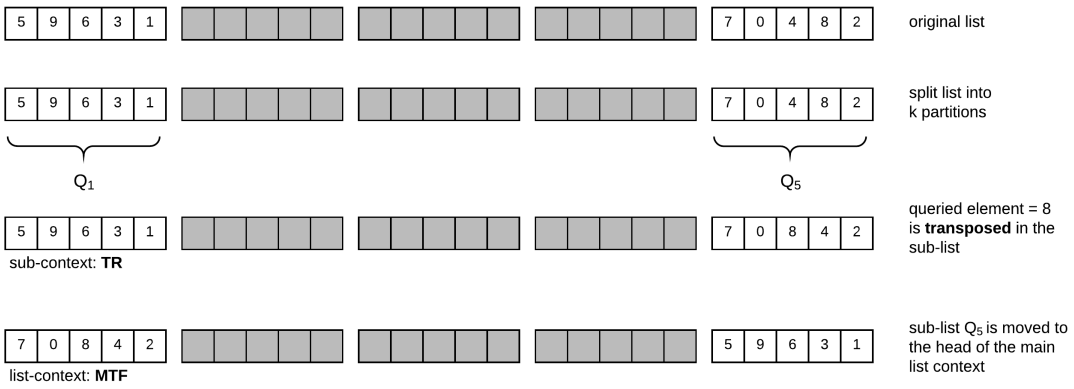


Figure 3.4: A diagrammatic description of the TR-MTF Hierarchical scheme

The algorithmic implementation for the TR-MTF is shown in Algorithm 18.

Algorithm 18 The TR-MTF Algorithm

Input:

- The list-on-list data structure with equi-partitioned elements, A.
- The stream of queries from the Environment, Q.

Output:

- The updated list using the TR-MTF rule.

1: **begin**
2: **repeat**
3: Receive queried element q_i
4: Get sub-list partition, k containing the query q_i
5: **if** index(q_i) ! $= 0$ **then** ▷ If q_i is not the list head
6: Swap element q_i with its preceding element, q_{i-1}.
7: **end if**
8: Move sub-list, k to the head of the list, A.
9: **until forever**
10: **end**

3.5.4 TR-TR

In the TR-TR hierarchical adaptive scheme, the accessed element within the sub-context is first moved one step towards the head of the sub-list, and then the entire sub-list is also promoted *en masse* one step towards the head of the main list. This rule is further illustrated in Figure 3.5. The algorithmic implementation for the TR-TR is shown in Algorithm 19.

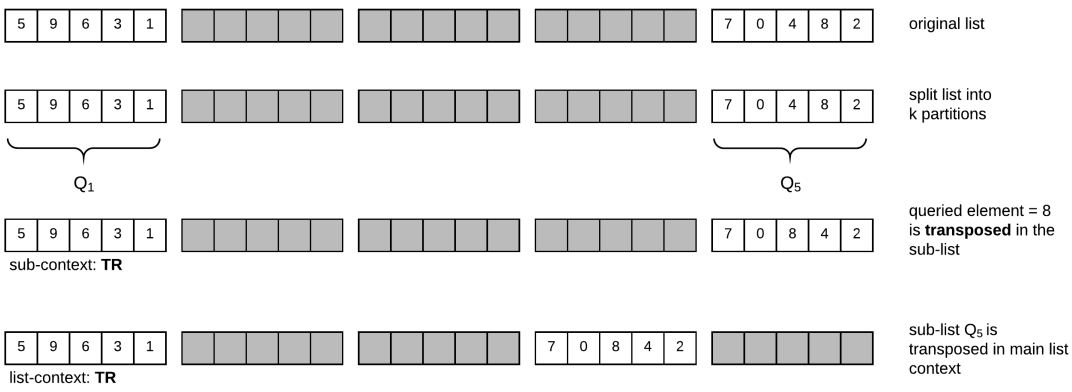


Figure 3.5: A diagrammatic description of the TR-TR Hierarchical scheme

Table 3.1 and Table 3.2 shows the comparison between the MTF and TR stand-alone reorganization rules and the hierarchical schemes in the MSEs and PSEs respectively. The simulation evaluated the amortized and asymptotic costs for an ensemble of 10 experiments, each running for $300,000$ query accesses. For the hierarchical lists, a list size of 128 records was divided into 8 sub-lists, each containing 16 records. The initial ordering of the list elements followed a uniform distribution. The simulations evaluated the performance of the adaptive schemes across a set of five Environments with different dependent Query Generators involving the Zipf, Eighty-Twenty, Lotka, Exponential and Linear query distributions.

From the simulation results in Table 3.1 and Table 3.2, we observe that the proposed hierarchical schemes performed worse than the stand-alone models in Environments with Locality of Reference. For example, in the MSE for the Zipf distribution, the MTF and TR rules yielded asymptotic costs of (43.35, 55.48) respectively, and amortized costs of (43.27, 55.59) respectively. Whereas in the case of the MTF-MTF,

Algorithm 19 The TR-TR Algorithm

Input:

- The list-on-list data structure with equi-partitioned elements, A.
- The stream of queries from the Environment, Q.

Output:

- The updated list using the TR-TR rule.

```
1: begin
2:     repeat
3:         Receive queried element q_i
4:         Get sub-list partition, k containing the query q_i
5:         if index(q_i) ! = 0 then                    ▷ If q_i is not the list head
6:             Swap element q_i with its preceding element, q_{i-1}.
7:         end if
8:         if index(k) ! = 0 then                      ▷ If k is not the list head
9:             Swap sub-list k with its preceding sub-list, k - 1.
10:        end if
11:    until forever
12: end
```

MTF-TR, TR-MTF and TR-TR, the schemes yielded asymptotic costs of (59.82, 59.66, 61.43, 60.97) and amortized costs of (59.82, 59.54, 61.37, 60.93) respectively. We thus observe that for all instances of the hierarchical schemes operating in Non-stationary Environments, the performance are worse than when one operates with the standalone schemes [3].

This observation is also true for all the other query distributions in both the Markovian and the Periodic Switching Environments. To record another example, consider the results for the Lotka distribution in the PSE, where the MTF and TR rule yielded asymptotic costs of (54.18, 48.47) and amortized costs of (54.16, 48.94) respectively. As opposed to this, for the MTF-MTF, MTF-TR, TR-MTF and TR-TR, the strategies yielded asymptotic costs of (58.54, 57.96, 58.11, 57.50) and amortized costs of (58.51, 57.97, 58.09, 57.52) respectively.

This phenomenon can be explained by the realization of the fact that the elements *within* a particular sub-list do not change so as to reflect the probabilistic dependence prescribed by the Environment. Thus, since the sub-lists are static, one can see that

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	43.35	43.72	39.12	8.68	43.62
TR	55.48	56.72	48.27	10.59	42.43
MTF-MTF	59.82	60.11	56.46	24.78	60.77
MTF-TR	59.66	59.84	55.97	20.47	60.51
TR-MTF	61.43	61.74	57.86	25.70	62.35
TR-TR	60.97	61.36	56.80	21.36	62.10
MTF	43.27	43.78	39.13	8.72	43.69
TR	55.59	57.04	48.67	10.96	57.37
MTF-MTF	59.82	60.23	56.46	24.85	60.67
MTF-TR	59.54	59.94	55.82	20.45	60.43
TR-MTF	61.37	61.75	57.84	25.66	62.28
TR-TR	60.93	61.44	56.88	21.34	62.08

Table 3.1: Experimental results between the hierarchical and stand-alone schemes in a Markovian Switching Environment with $\alpha = 0.9$. 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 8 sublists with 16 records each.

this will lead to an inferior access cost.

The problem associated with such static partitions is resolved by the introduction of the OMA family of LA partitioning algorithms that are able to capture the dependence of query requests from the Environment. They then partition elements possessing a strong dependence to each other within the same sub-list. This technique, of incorporating the OMA into the hierarchical schemes, is the subject of Chapter 4.

3.6 Concluding Remarks

In this Chapter, we introduced the concept of designing a list data-structure as a hierarchy of data "sub"-structures. It involved dividing a list of size J into k sublists. Thereafter, the re-organization strategy was applied first to the sub-contexts or

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	60.32	61.07	54.18	12.53	61.16
TR	55.69	56.85	48.47	11.72	57.16
MTF-MTF	62.09	62.52	58.54	25.77	62.87
MTF-TR	61.78	62.30	57.96	21.32	62.63
TR-MTF	61.73	62.15	58.11	27.07	62.73
TR-TR	61.60	62.08	57.50	22.72	62.51
MTF	60.25	61.06	54.16	12.59	61.21
TR	56.05	57.24	48.94	12.11	57.58
MTF-MTF	62.07	62.49	58.51	25.77	62.90
MTF-TR	61.83	62.30	57.97	21.39	62.65
TR-MTF	61.75	62.16	58.09	27.09	62.66
TR-TR	61.55	62.02	57.52	22.72	62.46

Table 3.2: Experimental results between the hierarchical and stand-alone schemes in a Periodic Switching Environment with a periodicity $T = 10$. 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 8 sublists with 16 records each.

elements within the sub-lists, and then to the sub-lists or list-context. The hierarchical schemes conceived were the MTF-preceding-MTF (MTF-MTF), MTF-preceding-TR (MTF-TR), TR-preceding-MTF (TR-MTF), and TR-preceding-TR (TR-TR) schemes.

However, from the simulation results reported in Table 3.1 and Table 3.2, that compared the performance of the hierarchical schemes with the standalone MTF and TR schemes in NSEs with Locality of Reference, we observed that the standalone schemes are superior to the hierarchical one, yielding lower access costs for both the Markovian and Periodic switching NSEs. This poor performance was due to the fact that the hierarchical schemes maintained a static grouping of the elements for all the sub-lists, thus leading to an inferior performance because the schemes were not able to reflect the probabilistic dependence between the queries.

We now proceed to show that we can mitigate this limitation by incorporating

the OMA family of reinforcement schemes from the theory of LA to handle the partitioning of elements with a strong probabilistic dependence into the same group. This enhancement and its corresponding improved results form the topic of Chapter 4

4

EOMA-Augmented Hierarchical SLLs

4.1 Introduction

This chapter presents the first novel results that we have developed. It is founded on incorporating an OMA-based scheme, explained earlier, into the theory of hierarchical lists strategies. The intention is to partition the elements within the sub-lists so that those that possess a strong probabilistic dependency gravitate together. We will first explain, very briefly, how the hierarchical schemes discussed in Chapter 3 can be augmented with the OMA algorithm to capture the dependence between the queries coming from the Environment. Such an "add-on" is intended to undo the static ordering of the sub-lists so that the elements can move freely from one sub-list partition to another.

The inclusion of the OMA as a preprocessor to the hierarchical schemes is not entirely novel - it was, rather, uniquely achieved by Amer and Oommen in [3]. With

the incorporation of the OMA algorithm, as in [3], we confirm that the hierarchical schemes, indeed, yield superior results when compared to the standalone MTF and TR schemes[1] in NSEs. This hypothesis is proven to be accurate in [3].

The OMA-incorporated algorithms that involve the primitive hierarchical schemes, give rise to the MTF-MTF-OMA, MTF-TR-OMA, TR-MTF-OMA and the TR-TR-OMA, where the third item in the triple is the reinforcement scheme in operation. In all brevity, this operated as follows. A higher level OMA was used to capture the dependence between the queries and take advantage of their Locality of Reference. This grouping, specified by the OMA, was then used to dictate the movement of the records within the sublist, and the migrations of the sublists themselves. This led to the four schemes alluded to above, namely, the MTF-MTF-OMA, MTF-TR-OMA, TR-MTF-OMA, and TR-TR-OMA. The details of each scheme and their results in NSEs, which were compared and contrasted with the standalone schemes, are found in [3].

The primary goal of this thesis is not intended to merely duplicate those results. That being said, we believe that it is wise to report them for a larger suite of datasets, and the latter results are reported in Appendix A. Our findings are that our newly reported results (in Appendix A) did not always concur with the results presented in [3]. This could be because of differences in the simulation parameters, ensemble settings, etc. However, the results presented in Appendix A, will at the very least, serve as a benchmark.

4.2 The EOMA *vs* the OMA

As mentioned earlier in Section 2.3.8, the basic OMA algorithm proposed in [3, 47] possesses a deadlock impediment. As discussed there, such a deadlock prevents the abstract objects from converging to their optimal partitioning. This deadlock issue was mitigated in the EOMA, as detailed in Section 2.3.8. Our goal in this chapter is to augment the hierarchical List-on-List SLLs with the EOMA.

[1] Observe that the standalone MTF and TR schemes are, in and of themselves, superior to the hierarchical schemes, as shown in Chapter 3.

From a naive perspective, it is easy to reckon that the effect of such a modification will be trivial. However, this is far from being true. For the first part, because of the deadlock scenario that the OMA, unfortunately, possesses, the accessed element can be swapped from one sublist to another and then back to the original sublist. Such an adverse occurrence has a further consequence because an unintended sublist could also be moved towards the front of the overall list structure. By mitigating such a deadlock scenario, the EOMA prohibits such "false alarm" swaps of elements between sublists, and also futile swaps between the various sublists themselves.

The second major issue that the EOMA takes care of involves the determination of the convergence criteria. Rather than wait "endlessly" for the accessed element to move into the most internal state of the OMA, the EOMA declares that the sublist has converged when the elements are within a few of the most internal states. This means that the EOMA is reasonably certain about the identity of the elements that should constitute a sublist, and does not permit a divergence from these sublists once it has declared that the sublists have converged.

But making sure that both of these are in place, one can obtain a superior *"en masse"* reorganization of the sublist groupings that allow us to minimize the query access costs. 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-TR-EOMA. As we shall presently demonstrate, they possess a superior performance, when it comes to minimizing the asymptotic and amortized query access costs over the OMA-augmented variants, for dependent Environments and for the models under consideration.

The following sections of this chapter report, in detail, the performance of these schemes, and additional simulation results are reported in Appendix B.

4.3 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 singly-list on singly-list 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 10.

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 20.

Algorithm 20 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.
- ξ 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.

4.4 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 10.

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 21[2].

4.5 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

[2]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 21 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.
- ξ 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.

penalty, the analysis operations are done as in Section 4.3 and 4.4, 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 22.

4.6 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 4.3, 4.4 and 4.5, 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 23.

4.7 Performance in MSEs

This section discusses the performance of the hierarchical schemes that incorporate the EOMA to manage the sub-list groupings so as to reflect the probabilistic dependency from the Environment. 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 sub-list, α, was set to 0.9. For all the results reported in this section, the simulation setup involved an ensemble of 10 experiments, each constituting 300,000 query accesses.

Algorithm 22 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.
- ξ 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 23 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.
- ξ 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.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	45.83	49.64	24.18	2.81	54.90
TR	37.43	41.51	19.01	2.48	48.36
MTF-MTF-EOMA	35.09	36.80	27.69	2.82	39.30
MTF-TR-EOMA	35.11	36.74	27.09	3.79	39.31
TR-MTF-EOMA	31.67	33.24	24.39	2.47	35.69
TR-TR-EOMA	31.60	33.21	26.06	2.55	35.76
MTF	45.79	49.63	24.10	2.82	54.93
TR	37.83	41.87	19.39	2.65	48.76
MTF-MTF-EOMA	35.22	36.70	27.04	2.81	39.27
MTF-TR-EOMA	35.21	36.85	27.69	3.80	39.30
TR-MTF-EOMA	31.87	33.43	25.55	2.55	35.86
TR-TR-EOMA	31.86	33.45	24.42	2.48	35.90

Table 4.1: 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**.

From the simulation results in Table 4.1, 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 Lokta 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.

To site an example, for the Linear distribution, the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA had asymptotic and amortized costs of (39.30, 39.31, 35.69, 35.76) and (39.27, 39.30, 35.86, 35.90) respectively compared to the stand-alone MTF and TR schemes which had asymptotic and amortized costs of (54.90, 48.36) and (54.93, 48.76) respectively. The reader can readily observe that the hierarchical results are superior to the stand-alone versions.

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 (2.82, 3.79, 2.47, 2.55) and (2.81, 3.80, 2.55, 2.48) respectively had comparable performance to the corresponding stand-alone MTF and TR schemes, with asymptotic and amortized costs of (2.81, 2.48) and (2.82, 2.65) respectively.

It is noteworthy that the Lotka distribution with asymptotic and amortized costs for the MTF and TR rules given as (24.18, 19.01) and (24.10, 19.39) are superior to those for the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA which are (27.69, 28.09, 24.39, 26.06) and (27.04, 27.69, 25.55, 24.42). 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.

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 4.2, 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 whose asymptotic and amortized costs for the MTF were still better than those obtained for the hierarchical schemes.

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.

By way of example, consider Table 4.2, 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 (18.27, 40.80, 18.07,

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	20.93	20.93	20.74	17.52	20.74
TR	60.36	60.40	58.17	36.66	59.87
MTF-MTF-EOMA	18.27	16.28	19.62	22.69	17.31
MTF-TR-EOMA	40.80	41.27	40.81	32.22	40.73
TR-MTF-EOMA	18.07	18.69	18.44	22.74	19.15
TR-TR-EOMA	41.01	41.44	40.99	32.64	41.04
MTF	20.94	20.95	20.77	17.60	20.72
TR	60.33	60.44	58.39	37.06	59.76
MTF-MTF-EOMA	19.65	18.28	20.63	22.81	18.87
MTF-TR-EOMA	40.86	40.97	40.58	32.20	40.88
TR-MTF-EOMA	19.37	19.92	20.25	22.69	16.43
TR-TR-EOMA	41.03	40.83	40.68	32.29	40.78

Table 4.2: 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**.

41.01) and amortized costs of (19.65, 40.86, 19.37, 41.03). As opposed to this, the stand-alone MTF and TR rule had asymptotic costs of (20.93, 60.36) and an amortized cost of (20.94, 60.33) respectively. These results corroborate the observation that the MTF-MTF-EOMA and the TR-MTF-EOMA schemes performed better than the MTF and TR schemes in all the dependent query Environments under consideration, whereas the MTF scheme was still superior to the MTF-TR-EOMA and the TR-TR-EOMA schemes.

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 4.1 - 4.4, 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, while results less

than unity show that the hierarchical EOMA scheme is superior to the MTF rule, which was the case which occurred most of the time.

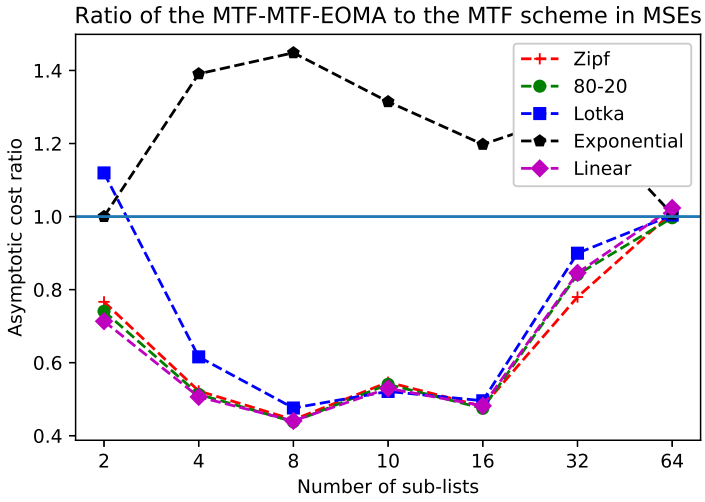


Figure 4.1: The asymptotic cost ratio of the MTF-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

From Figure 4.1, 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 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 more or less similar to that of the MTF-MTF-EOMA, as seen in Figure 4.2.

In Figure 4.3, 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

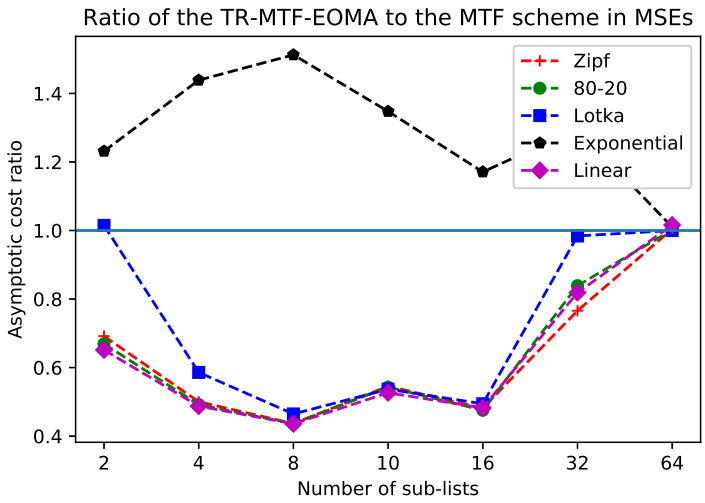


Figure 4.2: The asymptotic cost ratio of the TR-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

MTF was similar to that of the MTF-TR-EOMA in Figure 4.4 as they both had the TR as the outer-list reorganization rule.

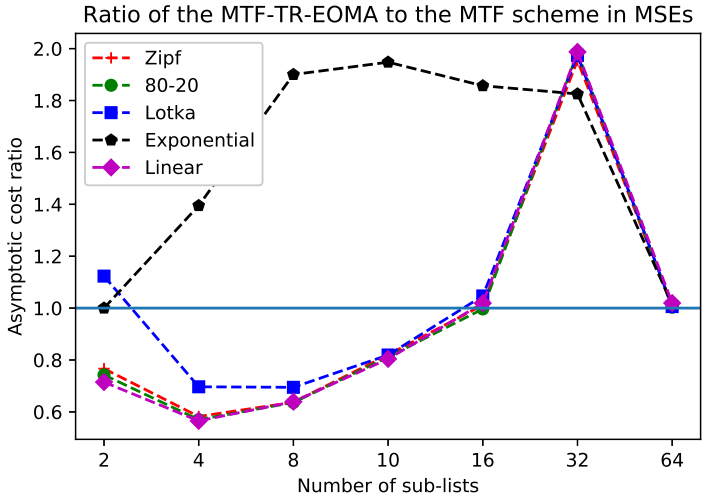


Figure 4.3: The asymptotic cost ratio of the MTF-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

To discuss the behavior of the hierarchical schemes which included the EOMA in

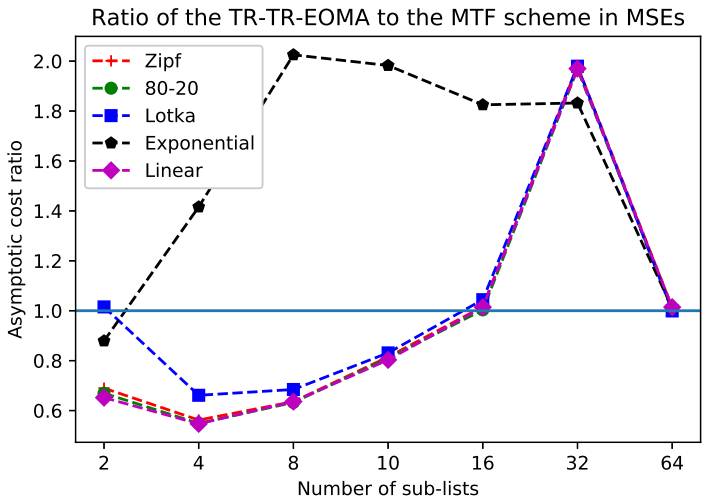


Figure 4.4: The asymptotic cost ratio of the TR-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

the MSEs, we considered their performance with respect to the Environments' degree of dependence. Observe that the "Locality of Reference" constant, α, controls the probabilistic measure that a query pair comes from the same query Environment or sub-list. Figure 4.5 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, $\alpha = 0.9$.

From the results in Figure 4.5, 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 α 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 4.5 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

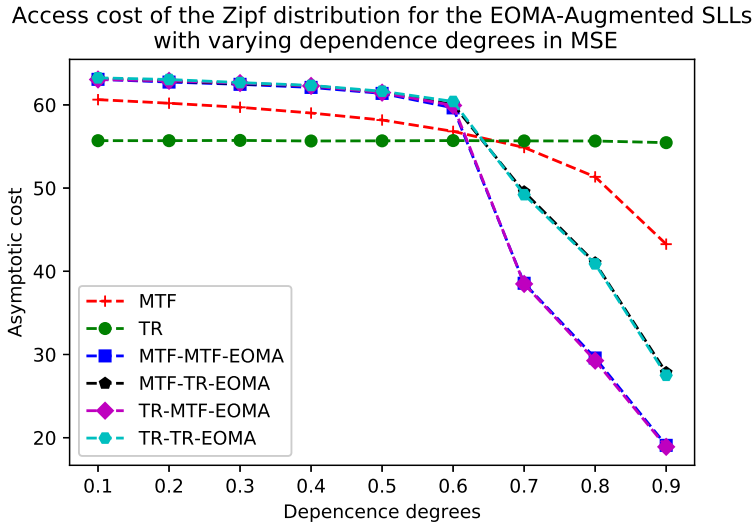


Figure 4.5: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with EOMA in the MSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

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 4.1 - 4.4).

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.

To display the rate of the convergence of the schemes under consideration, Figure 4.6 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 [37].

In Figure 4.6, 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

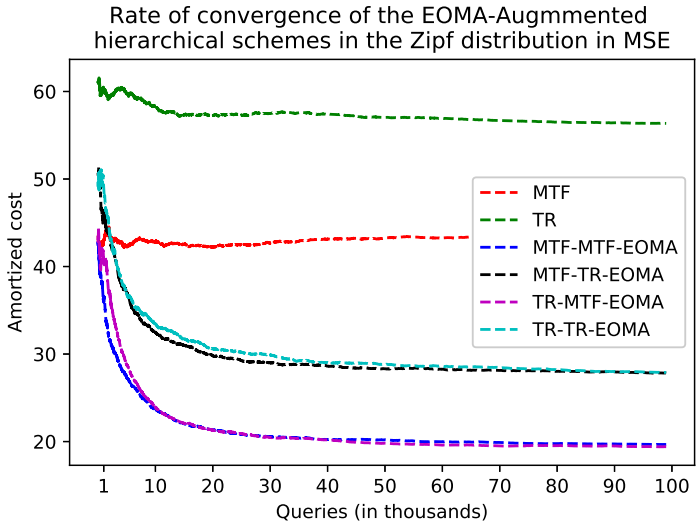


Figure 4.6: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes augmented with the EOMA in a MSE with $\alpha = 0.9$ and $k = 8$.

boasts a far superior performance compared to the MTF. A key observation is that the EOMA-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 4.6 shows the rate of convergence for the Zipf distribution, the observed phenomena were similar for the 80-20, Lotka, Exponential and Linear distributions.

4.8 Performance in PSEs

In this section, we report the results concerning the performance of the hierarchical schemes that incorporate the EOMA 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.

Table 4.3 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 4.3, 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 (29.96, 29.98, 26.41, 26.42) and (30.07, 32.90, 26.59, 26.61) respectively, while the MTF and TR schemes had an asymptotic and amortized cost of (50.66, 41.35) and (50.57, 41.88). Clearly, the EOMA-augmented hierarchical schemes were superior.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	46.61	50.66	24.27	2.52	56.21
TR	37.49	41.35	19.02	2.45	48.27
MTF-MTF-EOMA	28.33	29.96	20.33	2.84	32.47
MTF-TR-EOMA	28.28	29.98	20.35	2.51	32.44
TR-MTF-EOMA	24.80	26.41	17.66	2.43	28.94
TR-TR-EOMA	24.84	26.42	17.68	2.43	28.97
MTF	46.62	50.57	24.23	2.53	56.25
TR	37.92	41.88	19.42	2.62	48.75
MTF-MTF-EOMA	28.39	30.07	20.55	2.85	32.48
MTF-TR-EOMA	31.21	32.90	20.61	2.52	32.49
TR-MTF-EOMA	25.04	26.59	18.28	2.50	29.07
TR-TR-EOMA	25.07	26.61	18.20	2.50	29.11

Table 4.3: 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.

As opposed to this, for the Exponential distribution, the MTF and TR schemes had an asymptotic and amortized cost of (2.52, 2.45) and (2.53, 2.62), while the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA had corresponding costs of (2.84, 2.51, 2.43, 2.43) and (2.85, 2.52, 2.50, 2.50) respectively.

From this example, we see that the MTF and TR stand-alone schemes were comparable in performance to the hierarchical schemes that were augmented with the EOMA.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	18.01	18.01	17.89	16.45	17.74
TR	61.83	62.14	59.78	39.03	61.36
MTF-MTF-OMA	9.75	9.75	9.73	15.54	9.80
MTF-TR-OMA	66.33	66.32	66.30	22.28	66.20
TR-MTF-OMA	9.74	15.48	9.72	9.75	9.75
TR-TR-OMA	66.35	66.14	66.17	22.64	66.23
MTF	17.99	18.00	17.89	16.46	17.73
TR	62.01	62.31	60.11	39.40	61.61
MTF-MTF-OMA	11.01	11.06	10.98	16.79	11.02
MTF-TR-OMA	62.47	62.43	61.95	21.29	61.96
TR-MTF-OMA	10.87	16.57	11.16	10.97	10.90
TR-TR-OMA	62.28	62.10	61.96	21.64	62.21

Table 4.4: 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.

We now report the results for the scenario when the number of partitions increased, and $k = 32$. Table 4.4 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 (16.45, 39.03) and (16.46, 39.40) respectively. The asymptotic and amortized costs for the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and TR-TR-EOMA were (15.54, 22.28, 9.75, 22.64) and (16.79, 21.29, 10.97, 21.64) 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.

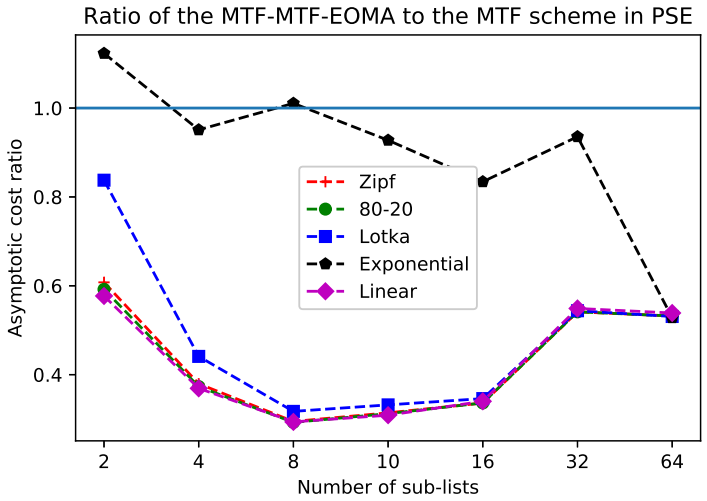


Figure 4.7: The asymptotic cost ratio of the MTF-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

We now report another perspective on the performance of the hierarchical schemes in PSEs as the number of sub-lists, k, increases. Figures 4.7 - 4.10 show the ratio of the asymptotic cost of the hierarchical schemes 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, and in the contrary case, if this index is less than unity, the respective hierarchical scheme is superior.

From Figure 4.7, 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 distribution in which case 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 4.8.

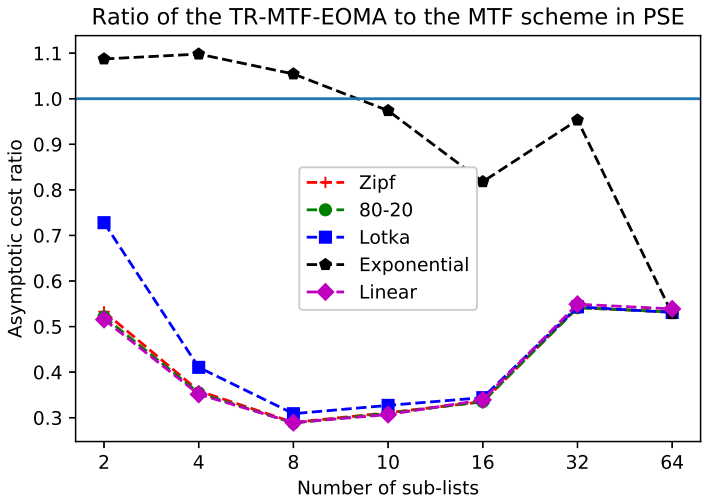


Figure 4.8: The asymptotic cost ratio of the TR-MTF-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

From Figure 4.9, 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 observation for the performance of the MTF-TR-EOMA against the MTF was similar to the TR-TR-EOMA displayed in Figure 4.10.

Figure 4.11 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.

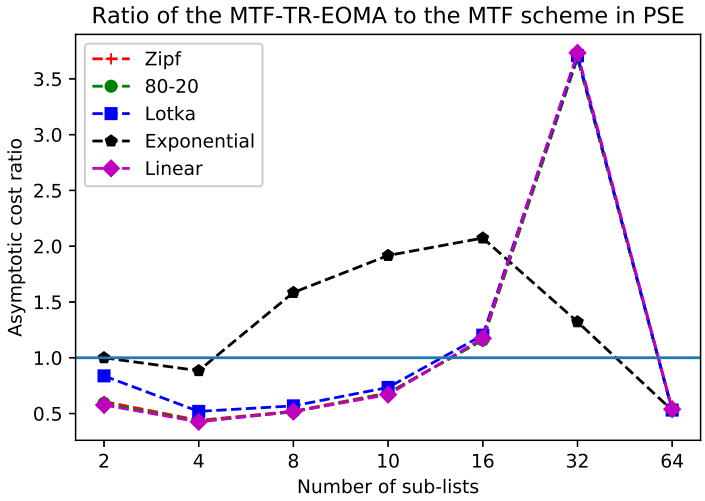


Figure 4.9: The asymptotic cost ratio of the MTF-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

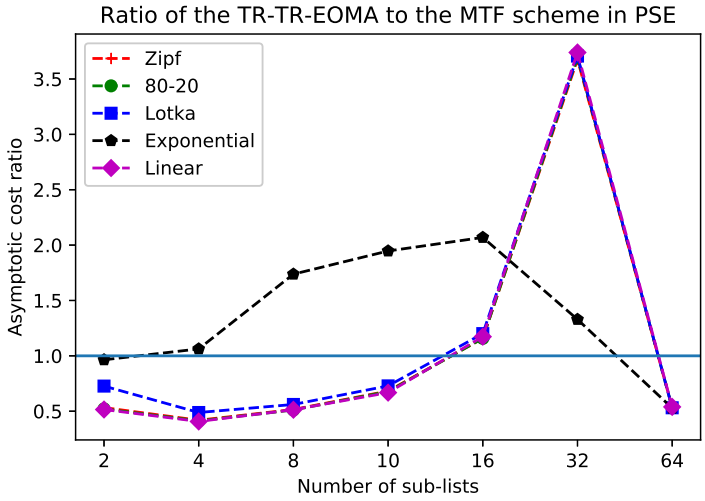


Figure 4.10: The asymptotic cost ratio of the TR-TR-EOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

From Figure 4.12 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

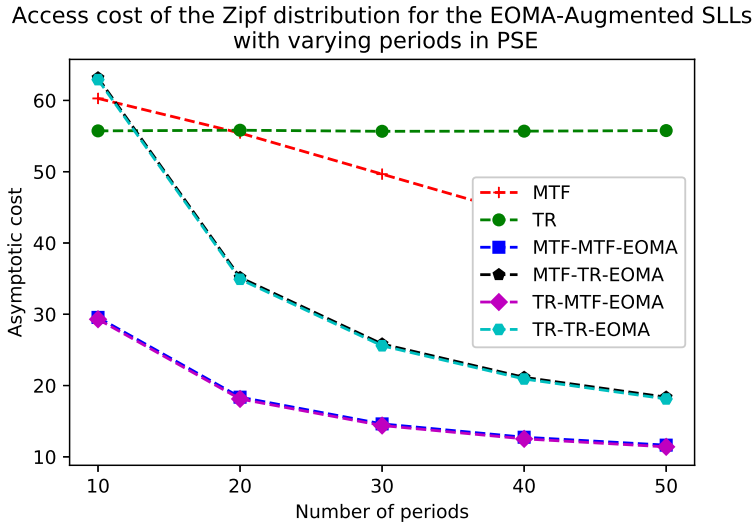


Figure 4.11: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with EOMA in the PSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

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.

4.9 Results for 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

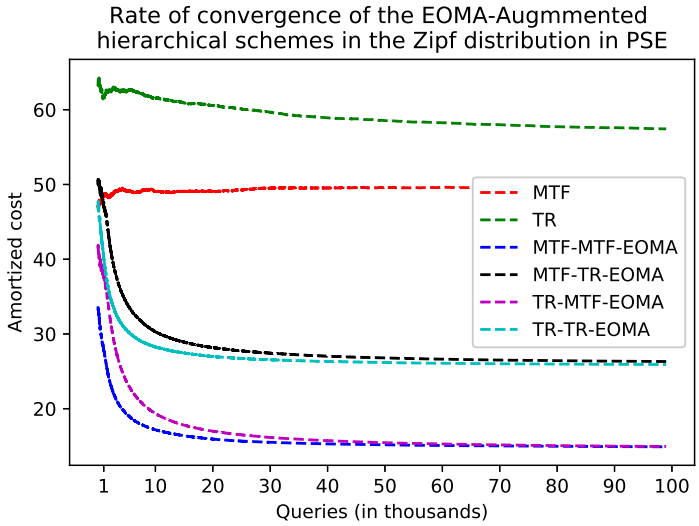


Figure 4.12: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with EOMA in the PSE with period $T = 30$ and $k = 8$.

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.

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-UnknownPeriod schemes. Figures 4.13 - 4.16 compare their relative performances against the regular hierarchical schemes that included the EOMA.

From Figure 4.13, 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 4.14.

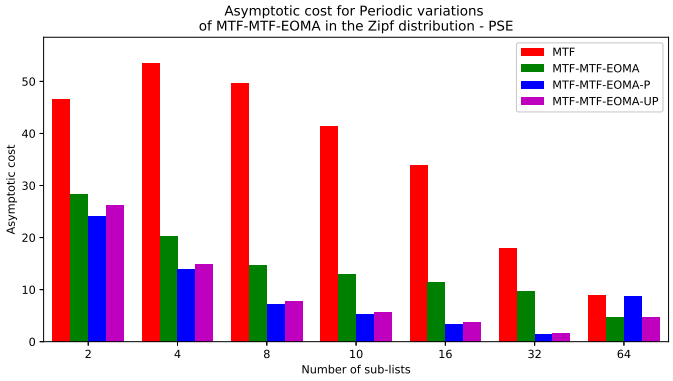


Figure 4.13: Asymptotic cost of Periodic variations of MTF-MTF-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

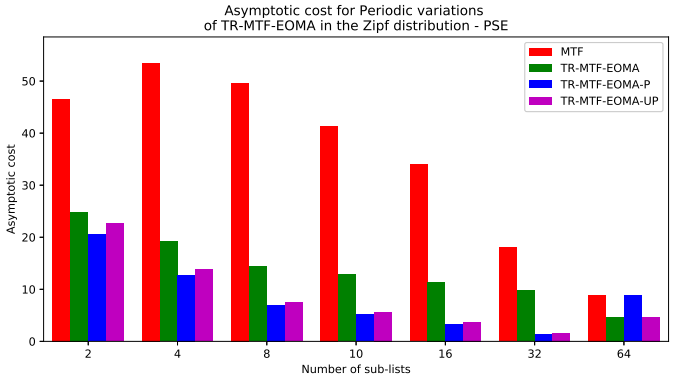


Figure 4.14: Asymptotic cost of Periodic variations of TR-MTF-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

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 4.16, had performances comparable to the MTF-TR-EOMA-Periodic/UnknownPeriod schemes. However, when $k = 64$, the TR-TR-EOMA-Periodic resulted in a surprisingly poor

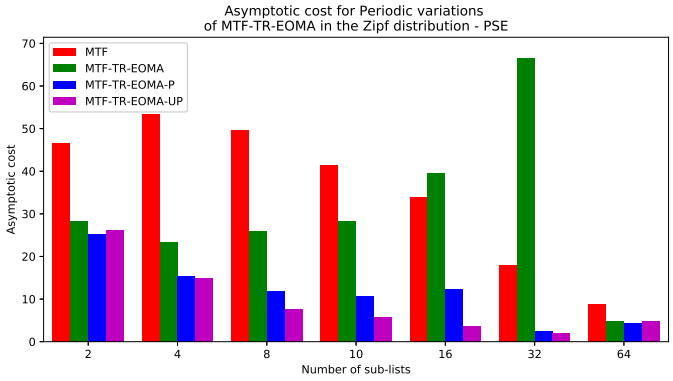


Figure 4.15: Asymptotic cost of Periodic variations of MTF-TR-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

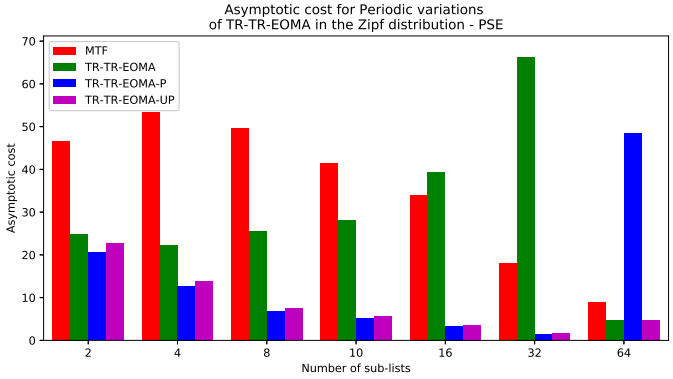


Figure 4.16: Asymptotic cost of Periodic variations of TR-TR-EOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

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, in which case the MTF and TR are identical strategies.

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

4.10 Concluding Remarks

In this chapter we incorporated the EOMA into the hierarchical schemes to break the static arrangement of their sublists. The EOMA enabled the hierarchical schemes to capture the probabilistic dependence ordering of the query accesses from the Environment. Further, the chapter discussed the performance of the MTF-MTF-EOMA, MTF-TR-EOMA, TR-MTF-EOMA and the TR-TR-EOMA for various sub-list values of k across the Zipf, 80-20, Lotka, Exponential and Linear distributions in the Markovian and Periodic Switching Environments.

The overall observation that we could make is that 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, was to an extent, 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[3]. Such a study remains open.

The following chapter will consider the case where the hierarchical schemes are enhanced with the Pursuit and Transitivity-augmented versions of the EOMA. The Pursuit concept improves the EOMA by equipping it with the ability to filter divergent queries from the Environment. Additionally, the Transitivity relationship is used to further infer good query pairs based on non-accessed elements that can improve the sub-list ordering of the List-on-Lists.

[3]We are very grateful to Professor Shirley Mills, one of the Examiners of the thesis, who suggested this.

5

PEOMA/TPEOMA-Augmented Hierarchical SLLs

5.1 Introduction

This chapter considers the case where the hierarchical SLL schemes are enhanced with the PEOMA and TPEOMA LA partitioning rules explained in Chapter 2. As mentioned earlier, the PEOMA incorporates the Pursuit concept to filter divergent queries from the Environment. As opposed to this, the Transitivity phenomenon, included in the TPEOMA, takes advantage of the statistical distribution of the queried elements to infer good query pairs from non-accessed elements in the transitivity relation. Both of these are used to improve the dependence-capturing aspect to be included in the sub-lists when it concerns the Lists-on-Lists hierarchy.

As in the previous chapter, the augmentation of hierarchical SLLs with the PEOMA

reinforcement scheme gives rise to the MTF-MTF-PEOMA, MTF-TR-PEOMA, TR-MTF-PEOMA and the TR-TR-PEOMA. Analogously, the TPEOMA-augmented hierarchical SLLsS result in the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and the TR-TR-TPEOMA.

The following sections of this chapter discuss the design and the performance of these schemes in the Markovian and Periodic Environments characterized by their respective models of dependence.

5.2 MTF-MTF-PEOMA

The algorithmic description of the PEOMA-augmented MTF-MTF scheme is shown in Algorithm 24. This algorithm makes use of the EOMA reinforcement strategy to update the MTF-MTF hierarchical scheme (Algorithm 25). The description of how the EOMA updates the MTF-MTF scheme was given in Section 4.3, and is not included here in the interest of avoiding repetition. Observe that the Pursuit phenomenon is specified in Lines 5 and 7, but the algorithm, otherwise, invokes the MTF-MTF-EOMA.

The difference between the MTF-MTF-PEOMA and the MTF-MTF-EOMA is that the former takes into account the addition of the Pursuit concept. This, in turn, utilizes the Pursuit matrix to collate the statistics used for filtering divergent queries.

5.3 MTF-TR-PEOMA

The MTF-TR-PEOMA algorithm incorporates the PEOMA reinforcement strategy to enhance the MTF-TR scheme. It operates as in Section 4.4, where the EOMA strategy was utilized to update the MTF-TR scheme. The formal algorithm is omitted to avoid repetition, but it is easily obtained by replacing Lines 8 and 11 in Algorithm 24 with a call to the EOMA-Augmented MTF-TR List Update. Again, the difference between the MTF-TR-EOMA and the MTF-TR-PEOMA schemes is the inclusion of the Pursuit concept.

Algorithm 24 The MTF-MTF-PEOMA 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.
- ξ is the states of the abstract objects O.
- A matrix of frequencies, $\mathcal{P}$, initially set to zeros.
- A user-defined threshold, τ, set to a value reasonable close to zero.
- A user-defined threshold, κ, to begin filtering queries.

Output:

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

```
1:  begin
2:      Initialization of {ξ_p}
3:      for a sequence of i ← 1, ..., T queries do
4:          Read query ⟨A_i, A_j⟩
5:          P[A_i, A_j] = P[A_i, A_j] + 1                ▷ Update the statistics
6:          if i > κ then
7:              if P[i, j]/ Σ_{i=1}^{W} P[i, j] ≥ τ then   ▷ Filter the Divergent queries
8:                  Call EOMA-Augmented MTF-MTF Update({ξ_p}, A_i, A_j)
9:              end if
10:         else
11:             Call EOMA-Augmented MTF-MTF Update({ξ_p}, A_i, A_j)
12:         end if
13:     end for
14: end
```

5.4 TR-MTF-PEOMA

In the TR-MTF-PEOMA, the TR-MTF hierarchical scheme is augmented with the PEOMA reinforcement scheme. Again, the PEOMA-augmented hierarchical scheme differs from its EOMA-augmented counterpart by the introduction of the Pursuit concept. The algorithmic description for the TR-MTF-PEOMA is obtained from Algorithm 24 by replacing Lines 8 and 11 with the EOMA-Augmented TR-MTF List Update.

Algorithm 25 EOMA-Augmented MTF-MTF List Update

Input:

- The physical elements $\{\langle A_p, A_q \rangle\}$ have corresponding abstract objects $\{O_1, \cdots, O_W\}$.
- ξ is the states of the abstract objects O.

Output:

- List-on-List updates according to the MTF-MTF-PEOMA rule.

1: **begin**
2: **if** ProcessReward($\{\xi_p\}, A_i, A_j$) **then** ▷ The partitioning is rewarded
3: Get sub-list partition, k containing the query A_j
4: Move element A_j to the head of sub-list, k.
5: Move sub-list k to the head of the list, A.
6: **else** ProcessPenalty($\{\xi_p\}, A_i, A_j$) ▷ The partitioning is penalized
7: **if** Both are in internal states **then**
8: Get sub-list k containing the query A_j
9: Move element A_j to the head of sub-list, k.
10: **else if** O_i is at boundary state **then**
11: Move element A_i to be in same sublist with A_j.
12: Move element A_k closest to boundary state of k_i to k_j.
13: **else if** O_j is at boundary state **then**
14: Move element A_j to be in same sublist with A_i.
15: Move element A_k closest to boundary state of k_j to k_i.
16: **else** Both are in boundary states
17: Remove A_j from sub-list k_j.
18: Get sub-list k_l containing the element l.
19: Remove l from sub-list k_l. ▷ Note that $k_l == k_i$
20: Append element l to sub-list k_j.
21: Append element A_j to sub-list k_i.
22: **end if**
23: **end if**
24: **end**

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

5.5 TR-TR-PEOMA

The TR-TR-PEOMA improves the TR-TR hierarchical scheme by using the PEOMA to capture dependent queries. The formal algorithm is obtained from Algorithm 24 by replacing Lines 8 and 11 with the EOMA-Augmented TR-TR List Update.

5.6 Performance of PEOMA-Hierarchical Schemes in MSEs

The experimental setup for the simulations involving the PEOMA-augmented hierarchical schemes in MSEs was the same as the one specified in Section 4.9, where a list of size 128 was split into k sublists, with $k \in \{2, 4, 8, 16, 32, 64\}$. The degree of dependence of the Environment, α, was also set to 0.9. For all the results discussed in this section, the simulation involved an ensemble of 10 experiments, each evaluating 300,000 query accesses, and for the various distributions mentioned earlier.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	45.83	49.64	24.18	2.81	54.90
TR	37.43	41.51	19.01	2.48	48.36
MTF-MTF-PEOMA	7.66	10.89	1.25	2.33	21.56
MTF-TR-PEOMA	7.71	10.83	1.24	3.24	21.59
TR-MTF-PEOMA	5.83	8.21	0.98	2.18	18.60
TR-TR-PEOMA	5.84	8.18	0.97	2.19	18.59
MTF	45.79	49.63	24.10	2.82	54.93
TR	37.83	41.87	19.39	2.65	48.76
MTF-MTF-PEOMA	10.47	14.36	2.61	2.37	23.47
MTF-TR-PEOMA	10.56	14.24	2.60	3.27	23.51
TR-MTF-PEOMA	8.63	11.41	3.42	2.26	20.34
TR-TR-PEOMA	8.62	11.41	3.00	2.27	20.37

Table 5.1: Experimental results displaying the average number of accesses for the hierarchical schemes that include the PEOMA 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**.

Consider Table 5.1, where we present the results when the PEOMA-augmented hierarchical scheme had a small sublist partition, $k = 2$. From this table we see that it provided a performance that was an order of magnitude superior to the standalone MTF and TR rules in both their asymptotic and amortized costs. This observation is true for all the models of dependence under consideration.

As an example, consider the Lotka distribution, which is one for which the MTF and TR standalone schemes were very competitive due to its L-shaped logarithmic curve that assigns higher probabilities to a small subset of the elements in the query space. The reader will recall that for this distribution, the EOMA-augmented hierarchical schemes found it difficult to compete with the MTF and TR rules, and was, for the most part, inferior to them with regard to their asymptotic and amortized costs.

However, with the PEOMA-augmented hierarchical schemes for the Lotka distribution, the results demonstrated that it was significantly superior to the standalone MTF and TR rules with asymptotic and amortized costs of the MTF-MTF-PEOMA, MTF-TR-PEOMA, TR-MTF-PEOMA and TR-TR-PEOMA being (1.25, 1.24, 0.98, 0.97) and (2.61, 2.60, 3.42, 3.00) respectively. In these cases, the MTF and TR had asymptotic and amortized costs of (24.18, 19.01) and (24.10, 19.39). The reader will observe that irrespective of the distribution shape that favours the MTF and TR rules, by incorporating the Pursuit concept into the EOMA to filter divergent queries, the sublist converges in a superior manner to model the query dependence ordering of the Environment. This is a fundamental contribution of this thesis

Now consider the case when the sublist partitions is larger, and $k = 32$. The reader will recall that in the Exponential distribution, the EOMA-enhanced hierarchical schemes which had the the outer-list context as the MTF, was superior to the standalone MTF and TR schemes. However, when the outer list context was the TR rule, the LOL performance was inferior to the MTF rule. However, in the PEOMA-enhanced versions, the schemes with TR as the outer list context was also more than an order of magnitude superior to the MTF and TR rules.

To illustrate this, consider the asymptotic and amortized costs for the MTF-MTF-PEOMA, MTF-TR-PEOMA, TR-MTF-PEOMA and TR-TR-PEOMA in the Exponential distribution with $k = 32$, which were (1.06, 1.15, 1.14, 3.18) and (3.00, 3.45, 3.13, 5.06) respectively. In these scenarios, the MTF and TR were (17.52, 36.66) and (17.60, 37.06). The superiority is remarkable!

That being said, we still see that for a hierarchical scheme augmented with an OMA-based strategy, when the outer-list context is the MTF, the performance is superior to the analogous scheme having the outer-list context as the TR.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	20.93	20.93	20.74	17.52	20.74
TR	60.36	60.40	58.17	36.66	59.87
MTF-MTF-PEOMA	1.73	1.50	1.49	1.06	1.24
MTF-TR-PEOMA	1.61	4.17	3.18	1.15	1.49
TR-MTF-PEOMA	1.62	1.59	3.79	1.14	1.35
TR-TR-PEOMA	1.49	1.47	4.42	3.18	1.29
MTF	20.94	20.95	20.77	17.60	20.72
TR	60.33	60.44	58.39	37.06	59.76
MTF-MTF-PEOMA	2.99	2.75	2.75	3.00	2.34
MTF-TR-PEOMA	3.31	5.83	3.19	3.45	3.21
TR-MTF-PEOMA	2.90	2.79	5.29	3.13	2.59
TR-TR-PEOMA	3.16	3.23	6.33	5.06	3.06

Table 5.2: Experimental results displaying the average number of accesses for the hierarchical schemes that included the PEOMA 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**.

Figures 5.1 to 5.4 display the ratio of the performances of the PEOMA-augmented hierarchical schemes to the MTF across sublist distributions from $k = 2$ to $k = 64$, and for all dependent model distributions in the MSE.

From Figure 5.1, we observed that the MTF-MTF-PEOMA was superior to the MTF for all distributions across all sublist partitions. This observation was also valid for the TR-MTF-PEOMA scheme as seen in Figure 5.2.

From Figure 5.3 we see that for the MTF-TR-PEOMA scheme, other than for the lone example when $k = 2$, the PEOMA-augmented schemes were progressively vastly superior to the MTF. For every other scheme under consideration, the MTF-MTF-PEOMA was an order of magnitude superior to the MTF. An identical observation can be made for the MTF-TR-PEOMA and TR-TR-PEOMA schemes, which are shown in Figure 5.4. However, the TR-TR-PEOMA scheme is marginally superior to the MTF-TR-PEOMA because it is superior to the MTF for all sublist partitions and models of dependence.

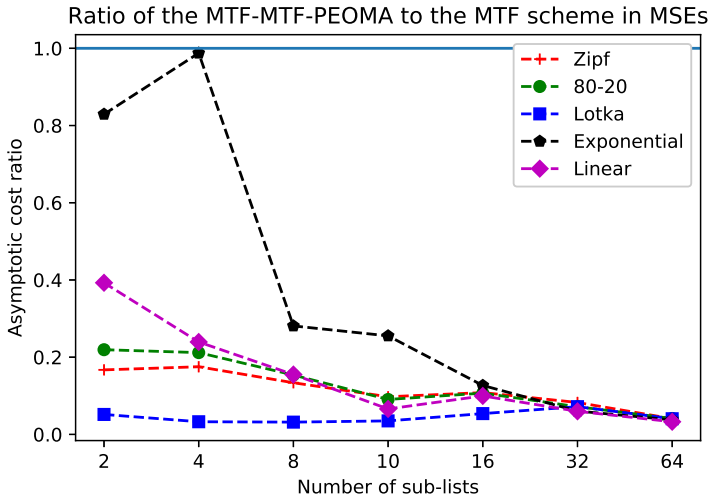


Figure 5.1: The asymptotic cost ratio of the MTF-MTF-PEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

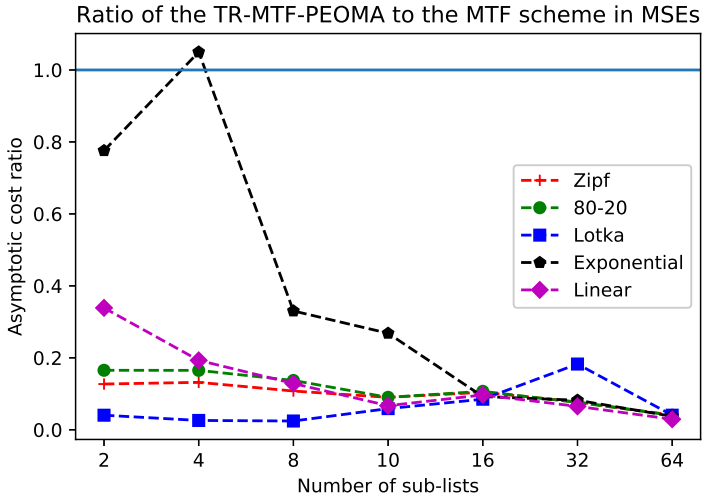


Figure 5.2: The asymptotic cost ratio of the TR-MTF-PEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

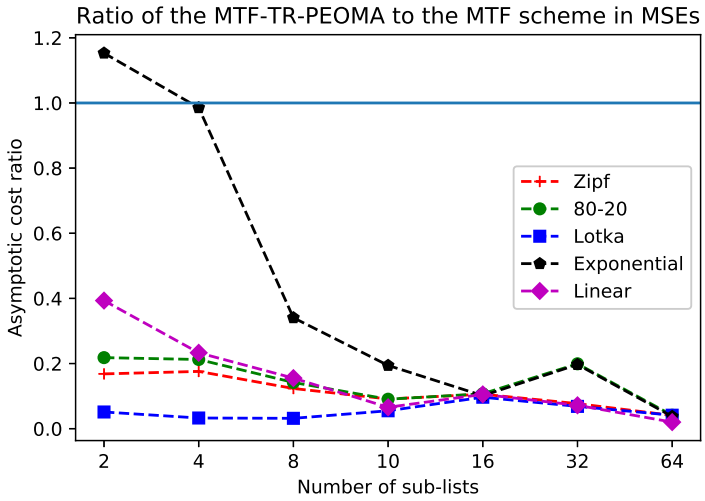


Figure 5.3: The asymptotic cost ratio of the MTF-TR-PEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

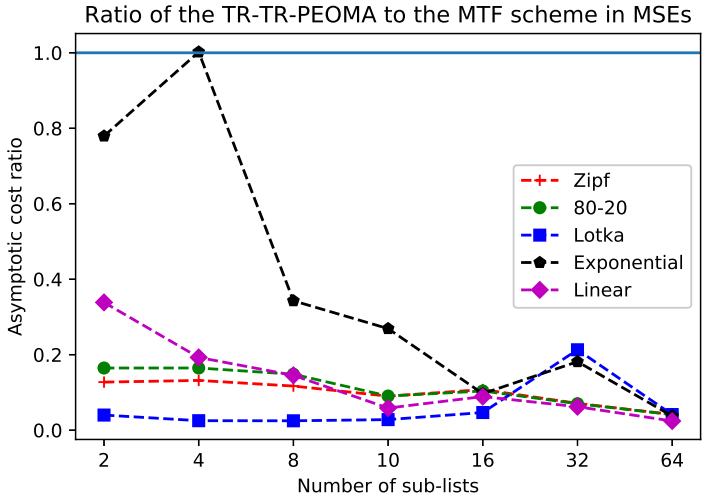


Figure 5.4: The asymptotic cost ratio of the TR-TR-PEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

Figure 5.5 displays the performance of the PEOMA-augmented hierarchical schemes as the degree of dependence in the MSE goes from a noisy Environment with $\alpha = 0.1$ to a less-noisy Environment with $\alpha = 0.9$. From Figure 5.5, we see that the performance of the PEOMA-augmented hierarchical schemes became increasingly superior to the MTF and TR rules when $\alpha > 0.2$. This performance was a marked improvement to the EOMA-augmented hierarchical schemes whose performance was superior to the MTF when $\alpha > 0.6$. This is another critical discovery reported in this thesis.

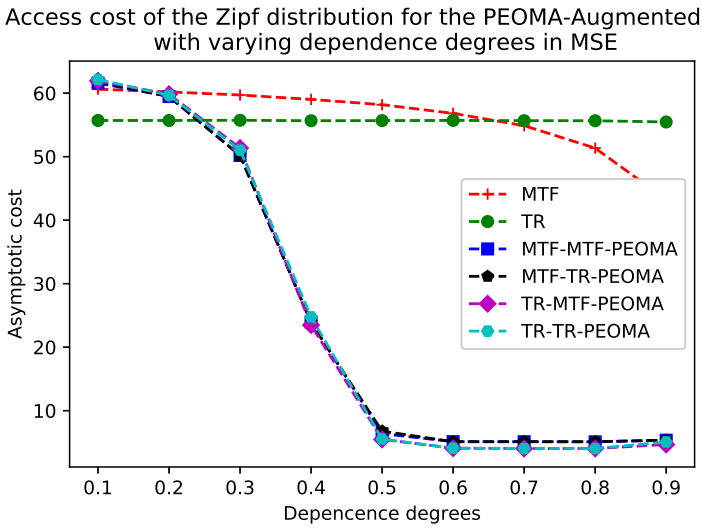


Figure 5.5: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with PEOMA in the MSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

From Figure 5.6, we observed that from approximately 1,000 queries, the PEOMA-augmented hierarchical schemes began to converge with an amortized cost superior to those provided by the MTF and TR rules. While the aforementioned standalone schemes plateaued in performance as they received more queries from the Environment, the cost associated with the PEOMA-augmented hierarchical schemes, on the other hand, were continuously decreasing, indicating that their performances became better in learning the true state of the querying Environment as the interactions increased.

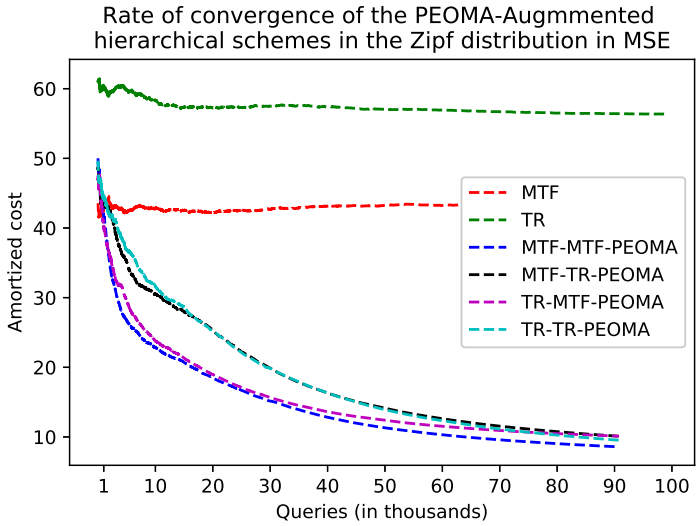


Figure 5.6: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with PEOMA in the MSE with $\alpha = 0.9$ and $k = 8$.

5.7 Performance of PEOMA-Hierarchical Schemes in PSEs

In order to access the performance of the PEOMA-augmented hierarchical schemes in the PSEs, the environmental setup mirrored the configuration used for the EOMA-augmented schemes in Section 4.8 with period $T = 30$.

From Table 5.3, when $k = 2$, we observed that the PEOMA-augmented hierarchical schemes were superior to the MTF and TR rules for all distributions under consideration. This performance is significant given that for the EOMA-augmented variant, the MTF and TR rules were competitive in the Exponential distribution. But this was not the case when divergent queries are filtered with the Pursuit matrix.

As an example to highlight this observation in the Exponential distribution, the asymptotic and amortized costs for the MTF-MTF-PEOMA, MTF-TR-PEOMA, TR-MTF-PEOMA and TR-TR-PEOMA were (2.16, 2.15, 2.52, 2.19) and (2.18, 2.19, 2.59, 2.26) respectively. The corresponding costs for the MTF and TR were (2.52, 2.45) and (2.53, 2.62) respectively.

Consider now the case where we had a large number of sublists, i.e., $k = 32$.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	46.61	50.66	24.27	2.52	56.21
TR	37.49	41.35	19.02	2.45	48.27
MTF-MTF-PEOMA	21.23	22.46	16.76	2.16	29.66
MTF-TR-PEOMA	21.36	22.79	16.86	2.15	29.58
TR-MTF-PEOMA	18.93	20.06	17.62	2.52	26.31
TR-TR-PEOMA	19.12	20.16	16.00	2.19	26.32
MTF	46.62	50.57	24.23	2.53	56.25
TR	37.92	41.88	19.42	2.62	48.75
MTF-MTF-PEOMA	22.39	24.23	17.44	2.18	30.58
MTF-TR-PEOMA	22.35	24.29	17.32	2.19	30.59
TR-MTF-PEOMA	19.99	21.57	18.68	2.59	27.33
TR-TR-PEOMA	20.00	21.58	16.70	2.26	27.32

Table 5.3: Experimental results between the hierarchical schemes with PEOMA 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.

The performance of the PEOMA-augmented hierarchical schemes were comparable to their EOMA-augmented variants. The hierarchical schemes with the outer-list context as the MTF had a superior performance to the MTF and TR rules. However, the schemes with TR as the outer-list context were inferior to all the other schemes. As an example, consider the Linear distribution when $k = 32$. The asymptotic and amortized costs for the MTF-MTF-PEOMA, MTF-TR-PEOMA, TR-MTF-PEOMA and TR-TR-PEOMA schemes were (9.77, 66.32, 9.77, 66.34) and (11.00, 62.44, 10.86, 62.26) respectively. In comparison, the costs for the MTF and TR rules were (17.74, 61.36) and (17.73, 61.61), confirming the above observation. As one can see, the MTF and TR rules were superior to the MTF-TR-PEOMA and TR-TR-PEOMA hierarchical schemes.

Figures 5.7 to 5.10 show the ratio of the asymptotic cost of the PEOMA-augmented hierarchical schemes to the MTF in the PSE. From Figure 5.7 we observed that the

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	18.01	18.01	17.89	16.45	17.74
TR	61.83	62.14	59.78	39.03	61.36
MTF-MTF-PEOMA	9.79	9.79	9.76	16.99	9.77
MTF-TR-PEOMA	66.34	66.34	66.31	19.78	66.32
TR-MTF-PEOMA	9.78	9.78	9.76	16.89	9.77
TR-TR-PEOMA	66.34	66.34	66.35	19.51	66.34
MTF	17.99	18.00	17.89	16.46	17.73
TR	62.01	62.31	60.11	39.40	61.61
MTF-MTF-PEOMA	11.03	10.88	11.07	17.39	11.00
MTF-TR-PEOMA	61.94	61.92	62.16	19.77	62.44
TR-MTF-PEOMA	11.09	11.00	11.12	17.41	10.86
TR-TR-PEOMA	62.46	62.18	62.09	19.97	62.26

Table 5.4: Experimental results between the hierarchical schemes with PEOMA 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.

performance of the MTF-MTF-PEOMA scheme was superior to the MTF for all distributions and across all sublists partitions except in the case of the Exponential distribution when $k = 32$, where we observed that the MTF was slightly inferior to the MTF-MTF-PEOMA in the PSE. The performance of the MTF-MTF-PEOMA was similar to the performance of the TR-MTF-PEOMA scheme.

From Figure 5.9, we observed that the MTF-TR-PEOMA performed better than the MTF for all distributions when $k < 16$ and $k = 64$. All the PEOMA-augmented hierarchical schemes had an inferior performance in comparison with the MTF when $k = 32$, as for the TR-TR-PEOMA scheme given in Figure 5.10

Figure 5.11 shows that the PEOMA-augmented hierarchical schemes in which the MTF was the outer-list context had a superior asymptotic cost to the MTF and TR rules as the period, T increased from 10 to 50. When the outer-list context was the TR, the hierarchical schemes was superior to the MTF when $T > 10$. The observation made from this figure was similar to the EOMA-augmented hierarchical schemes.

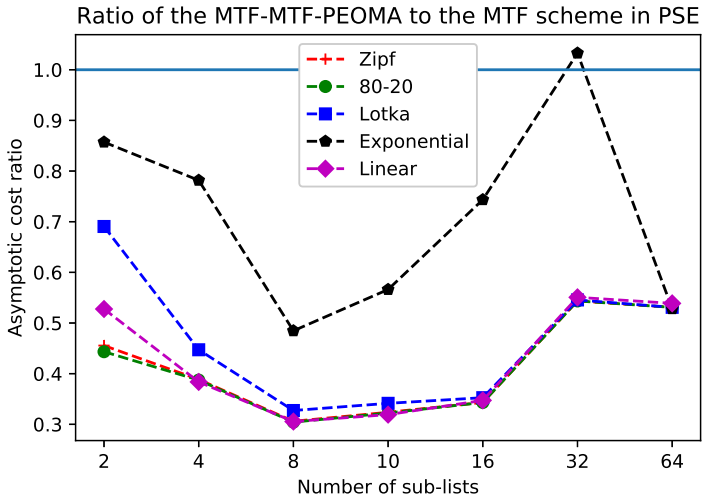


Figure 5.7: The asymptotic cost ratio of the MTF-MTF-PEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

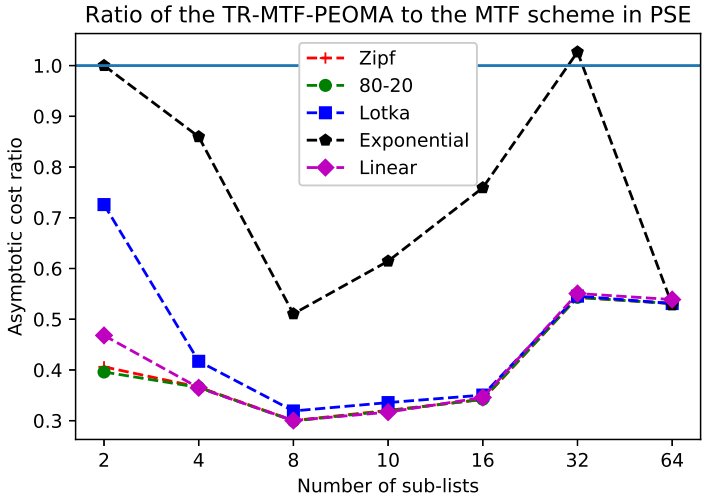


Figure 5.8: The asymptotic cost ratio of the TR-MTF-PEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

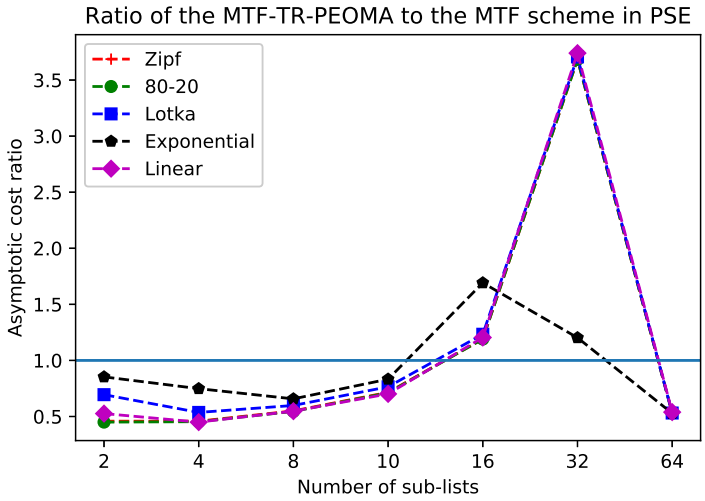


Figure 5.9: The asymptotic cost ratio of the MTF-TR-PEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

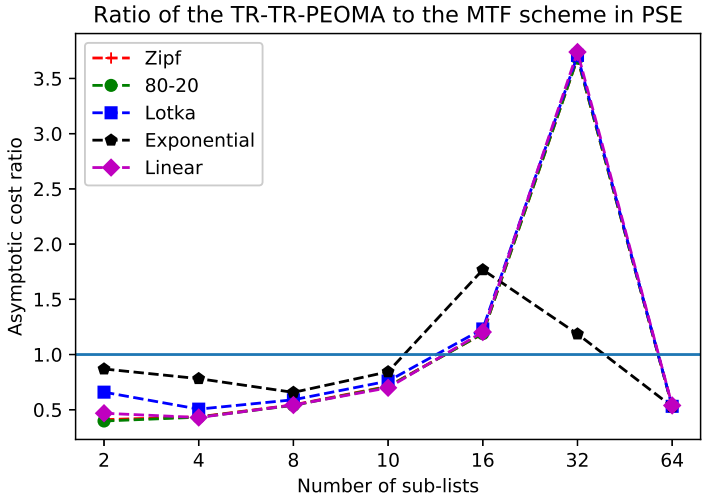


Figure 5.10: The asymptotic cost ratio of the TR-TR-PEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

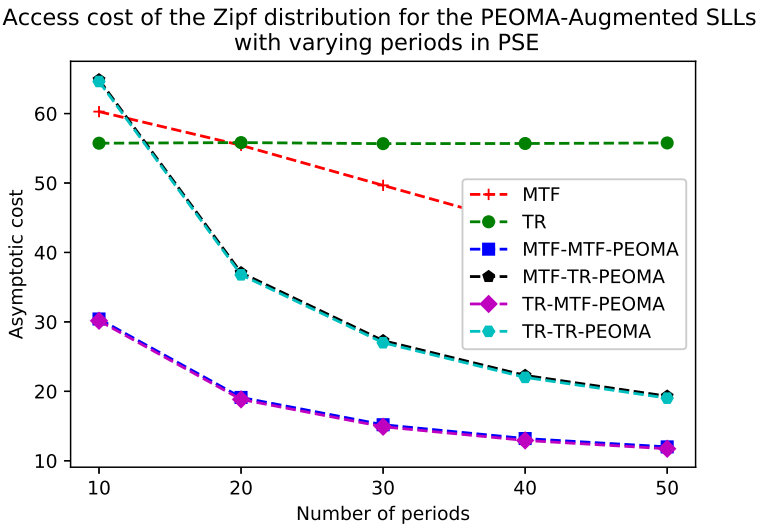


Figure 5.11: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with PEOMA in the PSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

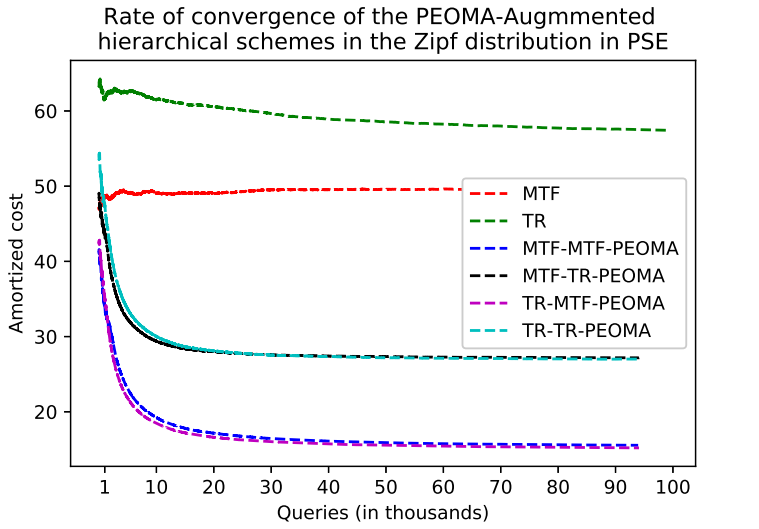


Figure 5.12: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with PEOMA in the PSE with period $T = 30$ and $k = 8$.

From Figure 5.12, we observed that the rate of convergence of the PEOMA-augmented hierarchical schemes with respect to the amortized costs were superior to the MTF and TR rules as soon as the LOL data-structure began to respond to queries from the Environment. The PEOMA-enhanced hierarchical schemes decreased rapidly and converges at about the $10,000^{th}$ query. Again, the performance was similar to that of the EOMA-augmented hierarchical schemes.

5.8 Results for Periodic variations with PEOMA-Hierarchical Schemes

This section reports the performance of the PEOMA-enhanced hierarchical schemes when the knowledge of the period change was incorporated into the algorithm. From the graphs in Figures 5.13 and 5.14, for the MTF-MTF-PEOMA and the TR-MTF-PEOMA schemes respectively, we see that the performance was generally superior to their vanilla versions and, indeed, the MTF standalone rule when the knowledge of the period was incorporated into the re-organization scheme.

For the MTF-TR-PEOMA and the TR-TR-PEOMA schemes displayed in Figures 5.15 and 5.16, when $k = \{16, 32\}$, we observed that the the "Periodic" and "Un-Periodic" versions of these schemes mitigated the poor performance of their vanilla versions against the MTF by displaying a vastly superior asymptotic cost to the MTF. As in the case of the EOMA-based algorithms, the "Periodic" versions of the MTF-TR-PEOMA and TR-TR-PEOMA when $k = 64$, performed poorly.

5.9 MTF-MTF-TPEOMA

The MTF-MTF-TPEOMA schemes incorporates the Transitivity relationship of the query pairs to improve the dependence clustering of the sublists. The details on the TPEOMA reinforcement scheme was discussed in Section 2.3.9. The difference between MTF-MTF-TPEOMA and MTF-MTF-PEOMA is the additional inclusion

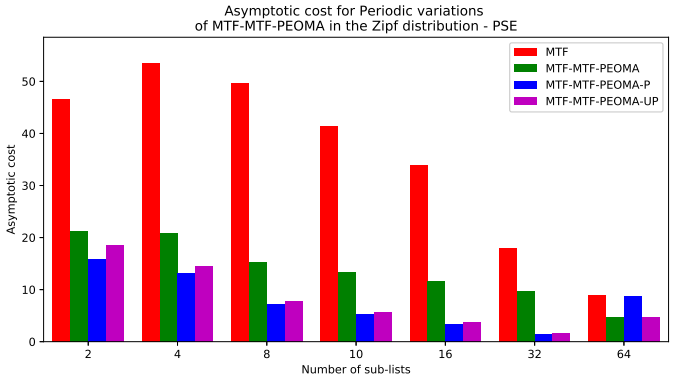


Figure 5.13: Asymptotic cost of Periodic variations of MTF-MTF-PEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

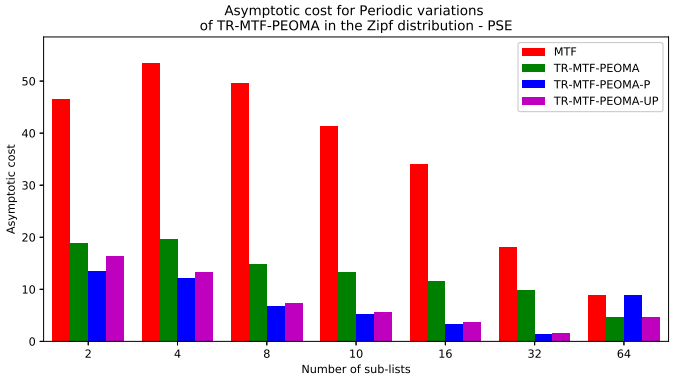


Figure 5.14: Asymptotic cost of Periodic variations of TR-MTF-PEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

of the Transitivity relationship that was able to infer good query pairs from non-accessed elements based on the statistical distribution of the queried elements. In the interest of completeness, the algorithmic description for the MTF-MTF-TPEOMA algorithm is shown in Algorithm 26. This algorithm, in turn, also makes use of the EOMA reinforcement strategy to update the MTF-MTF hierarchical.

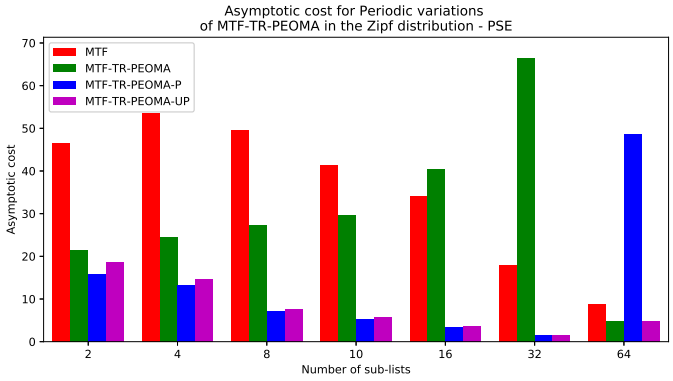


Figure 5.15: Asymptotic cost of Periodic variations of MTF-TR-PEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

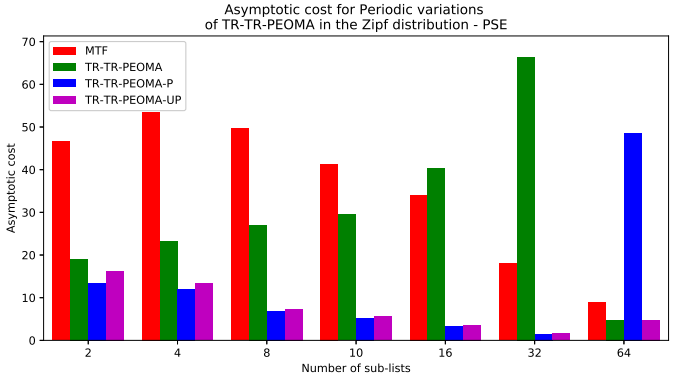


Figure 5.16: Asymptotic cost of Periodic variations of TR-TR-PEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

5.10 MTF-TR-TPEOMA

The MTF-TR-TPEOMA algorithm incorporates the TPEOMA reinforcement scheme to enhance the MTF-TR hierarchical scheme. Again, the formal algorithm is omitted to avoid repetition. It can in any case, be easily obtained by replacing Lines 10 and 13 in Algorithm 26 with a call to the EOMA-Augmented MTF-TR List Update.

Algorithm 26 The MTF-MTF-TPEOMA 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.
- ξ is the states of the abstract objects O.
- A matrix of frequencies, $\mathcal{P}$, initially set to zeros.
- A user-defined threshold, τ, set to a value reasonable close to zero.
- A user-defined threshold, κ, to begin filtering queries.

Output:

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

1: **begin**
2: Initialization of $\{\xi_p\}$
3: **for** a sequence of $i \leftarrow 1, ..., T$ queries **do**
4: Read query $\langle A_i, A_j \rangle$
5: $\mathcal{Z}[A_i, A_j] \leftarrow \mathcal{Z}[A_i, A_j] + 1$
6: $\mathcal{Z}[A_j, A_i] \leftarrow \mathcal{Z}[A_i, A_j]$
7: $\mathcal{P}_{i,j} \leftarrow \frac{\mathcal{Z}[A_i, A_j]}{\sum_{k,l=1}^{W} Z[A_k, A_l]}$ ▷ Update stats
8: **if** $i > \kappa$ **then**
9: **if** $\mathcal{P}[i,j] / \sum_{i=1}^{W} \mathcal{P}[i,j] \geq \tau$ **then** ▷ Filter the Divergent queries
10: Call EOMA-Augmented MTF-MTF Update($\{\xi_p\}, A_i, A_j$)
11: **end if**
12: **else**
13: Call EOMA-Augmented MTF-MTF Update($\{\xi_p\}, A_i, A_j$)
14: **end if**
15: **end for**
16: **end**

5.11 TR-MTF-TPEOMA

For the TR-MTF-TPEOMA, the TR-MTF hierarchical scheme is augmented with the TPEOMA reinforcement scheme. More precisely, the algorithmic description for the TR-MTF-TPEOMA is obtained by replacing Lines 10 and 13 in Algorithm 26 with a call to the EOMA-Augmented TR-MTF List Update.

5.12 TR-TR-TPEOMA

The TR-TR-TPEOMA augments the TR-TR hierarchical scheme by incorporating into it the TPEOMA reinforcement strategy. The formal algorithm for this procedure is again similar to Algorithm 26 after Lines 10 and 13 are replaced with a call to the EOMA-Augmented TR-TR List Update.

5.13 Performance of TPEOMA-Hierarchical Schemes in MSE

From Table 5.5, when $k = 2$, we observed that the TPEOMA-augmented hierarchical schemes were superior to the MTF and TR standalone schemes for all distributions except the Exponential distribution in which the MTF and TR possessed better asymptotic and amortized costs. As an example in the Lotka distribution, the asymptotic and amortized costs for the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA were (5.93, 5.94, 4.23, 4.41) and (7.42, 7.24, 6.44, 6.09) respectively. As opposed to this, the corresponding costs for the MTF and TR were significantly higher, i.e., (24.18, 19.01) and (24.10, 19.39) respectively.

Whereas, in the Exponential distribution where the standalone MTF and TR rules was superior, the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA had asymptotic and amortized costs of (5.91, 5.91, 3.93, 3.94) and (5.98, 5.95, 4.04, 4.09), while the MTF and TR rule had asymptotic and amortized costs of (2.81, 2.48) and (2.82, 2.65) respectively.

When the number of sublist increased, and $k = 32$, the TPEOMA-augmented hierarchical schemes all performed better than the MTF and TR rules for all the distributions of the Environment. Consider the Exponential distribution for which the MTF and TR rules had superior performances when $k = 2$. However, with $k = 32$, the asymptotic and amortized costs for the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA were (2.30, 2.45, 2.91, 1.68) and (4.37, 5.69, 4.84, 5.09), while those for the MTF and TR was (17.52, 36.66)

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	45.83	49.64	24.18	2.81	54.90
TR	37.43	41.51	19.01	2.48	48.36
MTF-MTF-TPEOMA	18.35	27.25	5.93	5.91	26.60
MTF-TR-TPEOMA	21.02	24.68	5.94	5.91	26.53
TR-MTF-TPEOMA	16.44	25.03	4.23	3.93	22.39
TR-TR-TPEOMA	18.77	26.60	4.41	3.94	22.38
MTF	45.79	49.63	24.10	2.82	54.93
TR	37.83	41.87	19.39	2.65	48.76
MTF-MTF-TPEOMA	22.20	29.73	7.42	5.98	27.56
MTF-TR-TPEOMA	22.82	28.43	7.24	5.95	27.52
TR-MTF-TPEOMA	22.56	31.44	6.44	4.04	23.33
TR-TR-TPEOMA	22.75	31.24	6.09	4.09	23.35

Table 5.5: Experimental results displaying the average number of accesses for the hierarchical schemes that include the TPEOMA 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**.

and (17.60, 37.06) respectively. The advantage gained is remarkable.

From Figures 5.17 to 5.20, we observed the asymptotic cost ratio of the TPEOMA-augmented hierarchical schemes to the MTF for varying number of sublist partitions ranging from $k \in 2, 4, 8, ..., 64$. Interestingly, in the case, the MTF-MTF-TPEOMA (Figure 5.17), TR-MTF-TPEOMA (Figure 5.18), MTF-TR-TPEOMA (Figure 5.18) and TR-TR-TPEOMA (Figure 5.20) all showed similar patterns for the Zipf, 80-20, Lotka, and Linear distributions, all performing better than the MTF for all sublist variations. However, for the Exponential distribution, the TPEOMA-augmented hierarchical schemes were superior to the MTF when $k > 4$.

From Figure 5.21, the TPEOMA-augmented hierarchical schemes outperformed the MTF and TR schemes in noisy Environment when the dependence degree $\alpha > 0.2$. Actually, when $\alpha = 0.2$, the TPEOMA enhanced schemes already displayed comparable (and in some cases better) performances to the MTF and TR schemes for the Zipf distribution.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	20.93	20.93	20.74	17.52	20.74
TR	60.36	60.40	58.17	36.66	59.87
MTF-MTF-TPEOMA	1.50	1.55	1.67	2.30	1.40
MTF-TR-TPEOMA	1.84	1.67	1.77	2.45	1.49
TR-MTF-TPEOMA	1.68	1.67	1.50	2.91	1.33
TR-TR-TPEOMA	1.49	1.68	1.68	1.68	1.50
MTF	20.94	20.95	20.77	17.60	20.72
TR	60.33	60.44	58.39	37.06	59.76
MTF-MTF-TPEOMA	3.39	3.58	3.56	4.37	3.24
MTF-TR-TPEOMA	4.31	4.28	4.38	5.69	3.83
TR-MTF-TPEOMA	3.71	3.62	3.44	4.84	3.16
TR-TR-TPEOMA	3.97	4.26	4.33	5.09	3.95

Table 5.6: Experimental results displaying the average number of accesses for the hierarchical schemes that included the TPEOMA 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**.

From Figure 5.22, we observed that from the first few queries accesses, the amortized cost of the TPEOMA-augmented hierarchical schemes begin to decrease, and displayed a continuously decreasing trend even after $90,000$ queries, indicating that the TPEOMA-enhanced LOL continued to learn as it interacted with the Environment.

5.14 Performance of TPEOMA-Hierarchical Schemes in PSEs

In the PSE, the TPEOMA-augmented hierarchical schemes did not provide us with the superior quality of results that we expected, if we compare it with what we observed for the PEOMA-enhanced schemes. Where we expected a similar or superior performance to the PEOMA-enhanced schemes, the TPEOMA-enhanced hierarchical

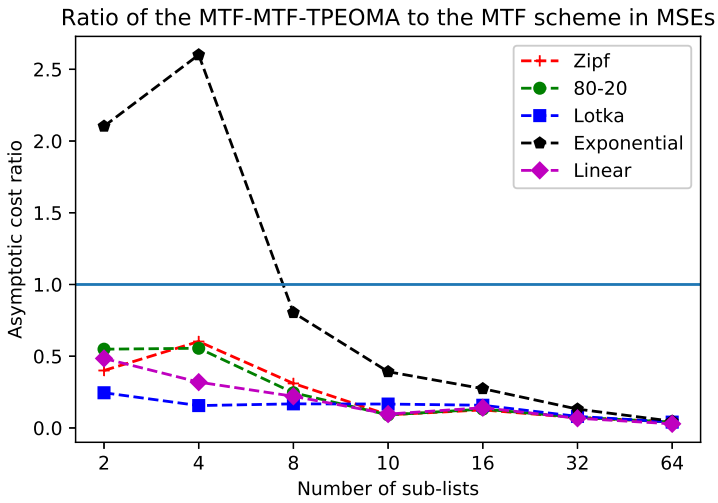


Figure 5.17: The asymptotic cost ratio of the MTF-MTF-TPEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

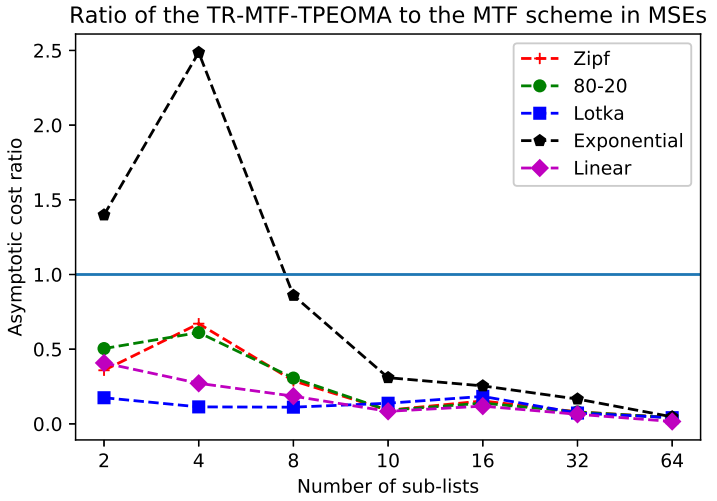


Figure 5.18: The asymptotic cost ratio of the TR-MTF-PEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

variants were otherwise surprisingly inferior. However, they still showed some good performances compared to the standalone MTF and TR schemes especially for the

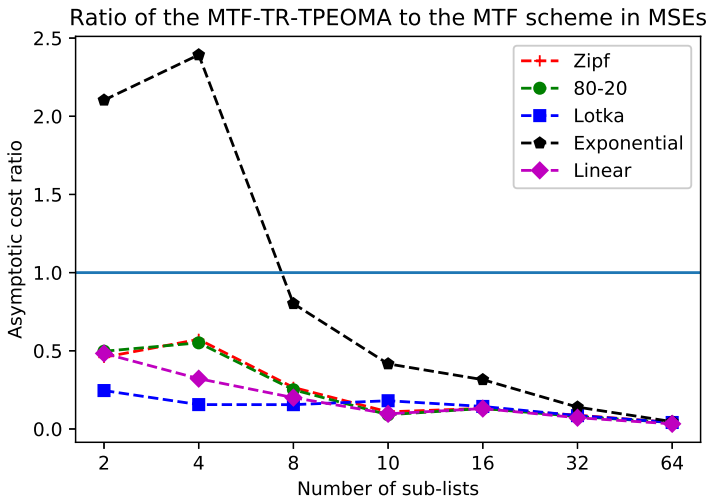


Figure 5.19: The asymptotic cost ratio of the MTF-TR-PEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

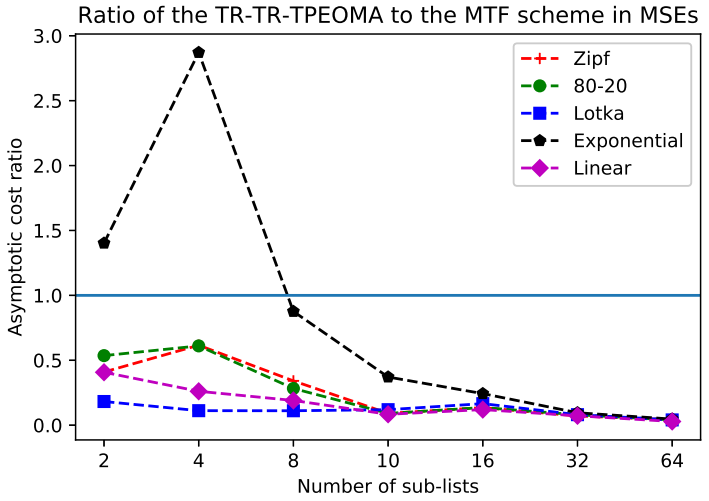


Figure 5.20: The asymptotic cost ratio of the TR-TR-PEOMA to the MTF scheme for different values of the sub-list partitions $k \in \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.

schemes where the MTF was the outer-list context. Quite honestly, we were rather disappointed because we expected more from the TPEOMA-augmented hierarchical

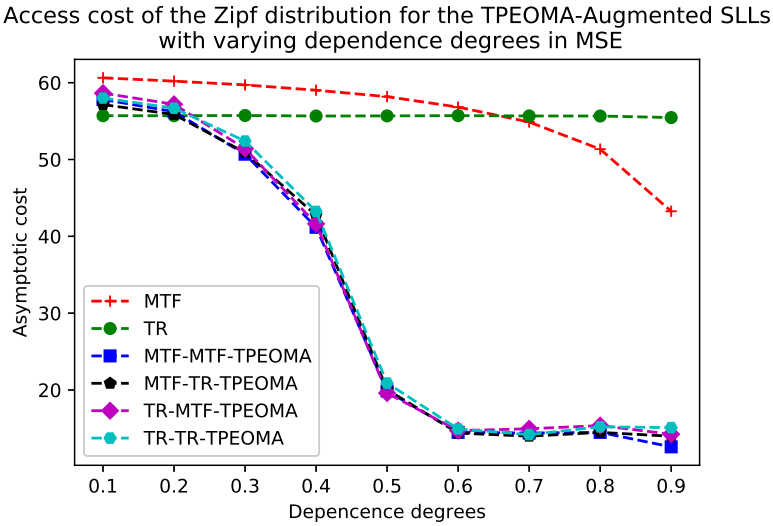


Figure 5.21: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with TPEOMA in the MSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

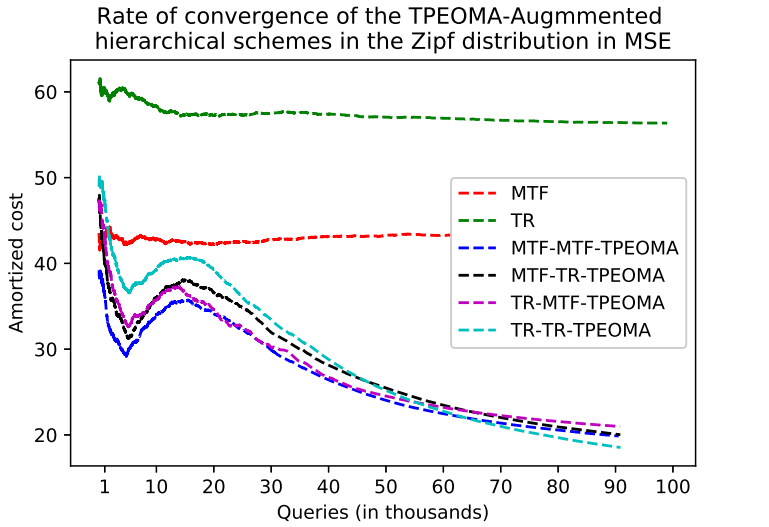


Figure 5.22: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with TPEOMA in the MSE with $\alpha = 0.9$ and $k = 8$.

schemes in the Periodic NSEs. In the interest of completeness, we report the results of the simulations.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	46.61	50.66	24.27	2.52	56.21
TR	37.49	41.35	19.02	2.45	48.27
MTF-MTF-TPEOMA	27.02	28.55	22.53	5.38	29.36
MTF-TR-TPEOMA	26.39	28.30	22.85	5.55	29.20
TR-MTF-TPEOMA	37.44	36.73	42.88	3.43	29.22
TR-TR-TPEOMA	37.67	36.05	43.16	3.46	29.24
MTF	46.62	50.57	24.23	2.53	56.25
TR	37.92	41.88	19.42	2.62	48.75
MTF-MTF-TPEOMA	28.56	30.70	22.42	5.42	29.97
MTF-TR-TPEOMA	28.23	30.82	22.50	5.53	29.93
TR-MTF-TPEOMA	37.74	39.06	41.49	3.51	28.73
TR-TR-TPEOMA	37.81	38.62	41.62	3.52	28.77

Table 5.7: Experimental results between the hierarchical schemes with TPEOMA 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.

From Table 5.7, when $k = 2$, the TPEOMA-augmented hierarchical schemes achieved superior performances over the MTF and TR standalone schemes for the 80-20 and the Linear distributions. In the Zipf and Lotka distributions, the MTF-MTF-TPEOMA and the MTF-TR-TPEOMA was superior to the MTF and TR, However, the performance of TR-MTF-TPEOMA and the TR-TR-TPEOMA, while superior to the MTF, had comparable results with the TR rule. For the Exponential distribution, the MTF and TR had superior asymptotic and amortized costs.

As an example, consider the 80-20 distribution where the TPEOMA-augmented hierarchical schemes were superior. The asymptotic and amortized costs for the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA were (28.55, 28.30, 36.73, 36.05) and (30.70, 30.82, 39.06, 38.62) while those for the MTF and TR rules were markedly worse, i.e, (50.66, 41.35) and (50.57, 41.88). As opposed to this, in the Exponential distribution, the MTF and TR schemes were superior. Thus, the asymptotic and amortized costs for the MTF-MTF-TPEOMA,

MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA were (5.38, 5.55, 3.43, 3.46) and (5.42, 5.53, 3.51, 3.52) while those for the MTF and TR rule were (2.52, 2.45) and (2.53, 2.62) respectively.

Scheme	Zipf	80-20	Lotka	Exponential	Linear
MTF	18.01	18.01	17.89	16.45	17.74
TR	61.83	62.14	59.78	39.03	61.36
MTF-MTF-TPEOMA	14.84	14.85	14.67	11.65	14.74
MTF-TR-TPEOMA	32.31	32.25	32.31	28.30	32.03
TR-MTF-TPEOMA	14.75	14.76	14.61	11.75	14.79
TR-TR-TPEOMA	32.13	32.15	31.91	28.30	31.67
MTF	17.99	18.00	17.89	16.46	17.73
TR	62.01	62.31	60.11	39.40	61.61
MTF-MTF-TPEOMA	14.73	14.78	14.58	11.81	14.68
MTF-TR-TPEOMA	31.97	32.08	31.99	28.41	31.67
TR-MTF-TPEOMA	14.79	14.78	14.61	11.84	14.71
TR-TR-TPEOMA	32.02	31.97	31.87	28.39	31.62

Table 5.8: Experimental results between the hierarchical schemes with TPEOMA 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.

From Table 5.8, where the number of sublist partitions increased to $k = 32$, the TPEOMA-augmented hierarchical schemes which had the MTF as its outer-list context yielded a superior performance to the MTF and TR rules across all distributions. However, when the outer-list context was the TR rule, the TPEOMA enhanced hierarchical schemes was inferior to the MTF across all distributions. Based on the previous results, this phenomenon was as expected.

As an example, consider the Exponential distribution. For $k = 32$, the asymptotic and amortized costs of the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA schemes were (11.65, 28.30, 11.75, 28.30) and (11.81, 28.41, 11.84, 28.39) respectively. However, those of the MTF and TR

schemes were (16.45, 39.03) and (16.46, 29.40) respectively. The results further highlight the observation that the schemes with the MTF as the outer-list context were the preferred hierarchical schemes to be used in Periodic NSEs.

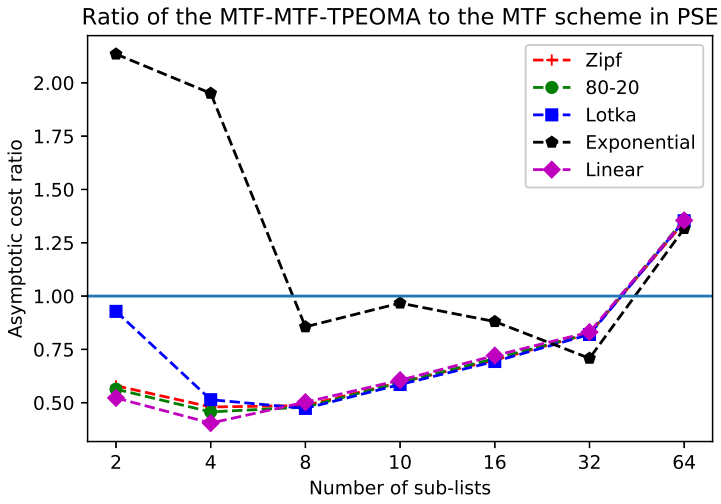


Figure 5.23: The asymptotic cost ratio of the MTF-MTF-TPEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

Figures 5.23 to 5.27 illustrate the performance ratio of the asymptotic costs for the TPEOMA-augmented hierarchical schemes to the MTF rule. From Figure 5.23, we noted that the MTF-MTF-TPEOMA had superior performances to the MTF for the Zipf, 80-20, Lotka and Linear distributions where $k = \{2, 4, 6, 10, 16, 32\}$. In the Exponential distribution, the MTF-MTF-TPEOMA was only superior to the MTF when $k = \{8, 10, 16, 32\}$. When $k = 64$, the MTF was superior to the hierarchical schemes with TPEOMA. The performance of the MTF-MTF-TPEOMA was roughly similar to that of the TR-MTF-TPEOMA shown in Figure 5.24.

From Figure 5.25, the MTF-TR-TPEOMA has superior performance to the MTF for the Zipf, 80-20, Lotka and Linear distribution when $k = \{2, 4, 8, 10\}$. When $k = 16$, their results were comparable, and when $k > 16$, the MTF was superior. In the Exponential distribution, the MTF-TR-TPEOMA was superior to the MTF only when $k = 8$. The result of the MTF-TR-TPEOMA was similar to the TR-TR-TPEOMA scheme, as shown in Figure 5.26.

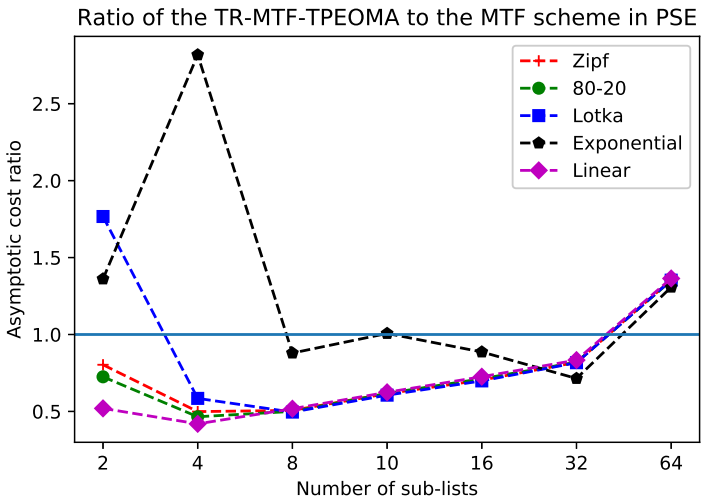


Figure 5.24: The asymptotic cost ratio of the TR-MTF-TPEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

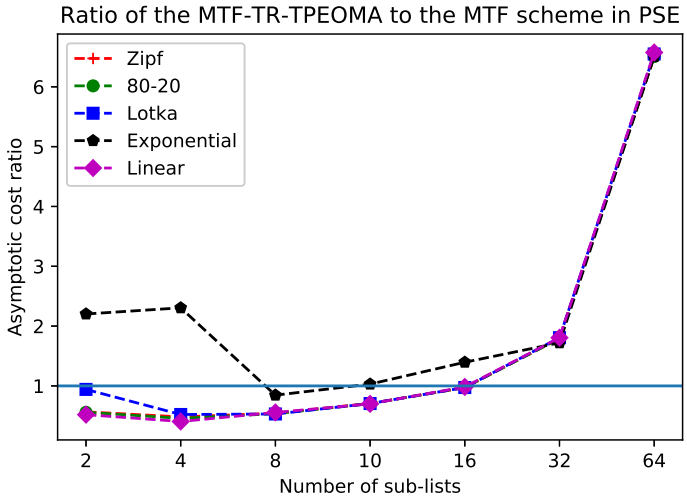


Figure 5.25: The asymptotic cost ratio of the MTF-TR-TPEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

Figure 5.27 shows the asymptotic costs for the TPEOMA-augmented hierarchical schemes across varying number of periods for the Zipf distribution. From Figure 5.27,

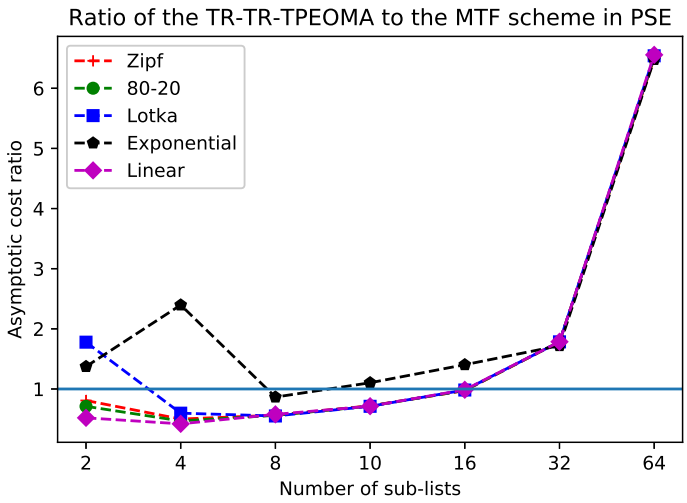


Figure 5.26: The asymptotic cost ratio of the TR-TR-TPEOMA to the MTF scheme for different values of the sub-list partitions $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$ across the dependent query Environments of the PSE with period $T = 30$.

we observed that the performance of the TPEOMA-augmented hierarchical schemes improved when the periodicity, T, increased. This was because the TPEOMA reinforcement scheme was able to better learn the dependence grouping of elements before the query Environment changed.

Figure 5.28 shows the rate of convergence of the TPEOMA-augmented hierarchical schemes in the Zipf distribution. The key observation from this was that the TPEOMA enhanced scheme quickly converged with the first few queries, and was superior to the MTF and TR standalone schemes.

5.15 Results for Periodic variations with TPEOMA-Hierarchical Schemes

The "Periodic" and "UnPeriodic" variations of the TPEOMA-augmented hierarchical schemes were generally unstable, and from our experiments perform worse than their vanilla counterparts and in many cases is inferior to the MTF scheme. The results of incorporating the knowledge of the period at which the query space changed are shown

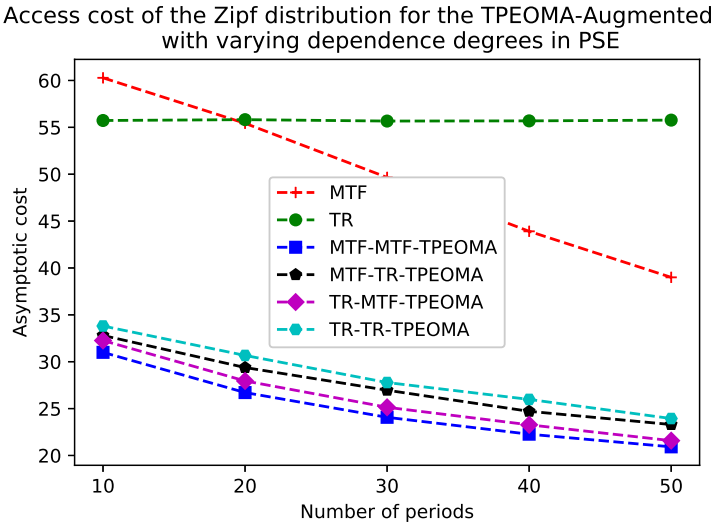


Figure 5.27: Changes in the asymptotic cost of the stand-alone and hierarchical schemes with TPEOMA in the PSE. In this experiment, a list of 128 elements is partitioned into 8 sub-lists.

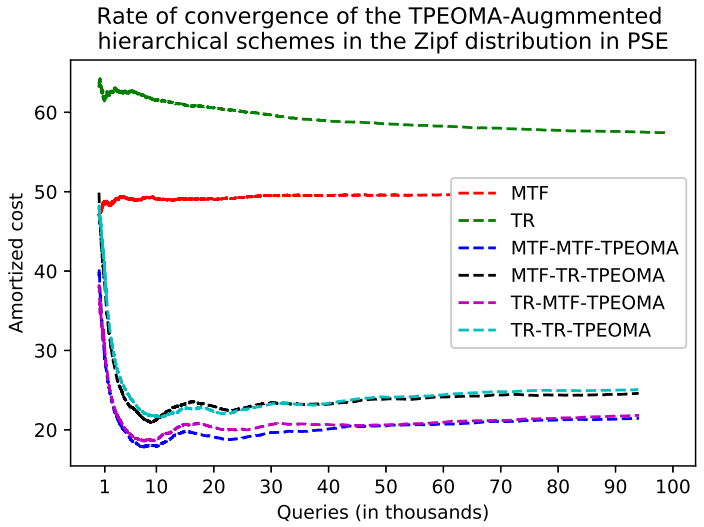


Figure 5.28: Rate of convergence of the first 100,000 queries for the stand-alone and hierarchical schemes with TPEOMA in the PSE with period $T = 30$ and $k = 8$.

in Figures 5.29, 5.30, 5.31, and 5.32 respectively for the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA.

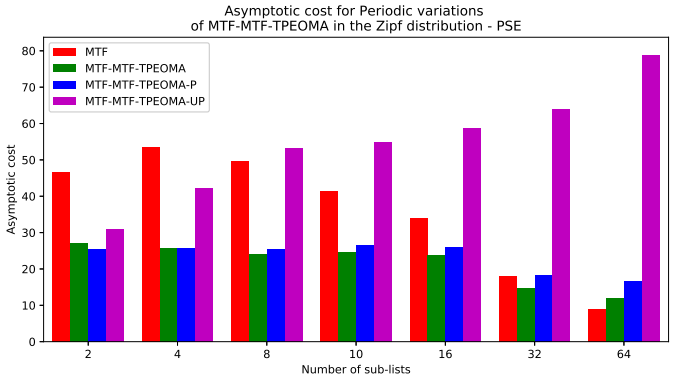


Figure 5.29: Asymptotic cost of Periodic variations of MTF-MTF-TPEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

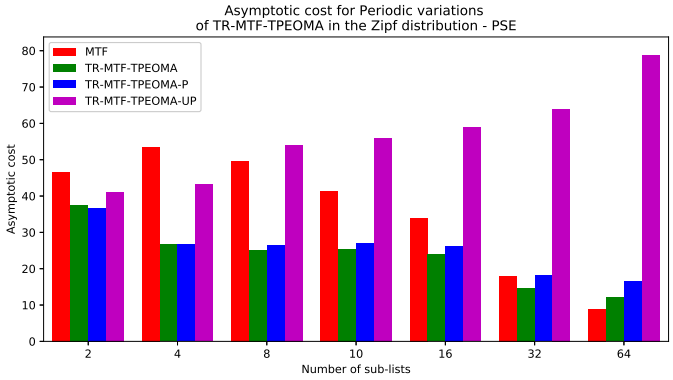


Figure 5.30: Asymptotic cost of Periodic variations of TR-MTF-TPEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

5.16 Concluding Remarks

In this chapter, we incorporated the PEOMA and TPEOMA reinforcement strategies to augment the SLLs hierarchical schemes. The results for the PEOMA-augmented hierarchical schemes showed an order of magnitude superior performance in the MSE Environments when compared to the standalone MTF and TR rules. Also in the PSE Environment, the PEOMA-augmented hierarchical schemes also boasted superior performances. When the knowledge of the query periods was incorporated into the PEOMA-augmented hierarchical schemes, the results were greatly improved to

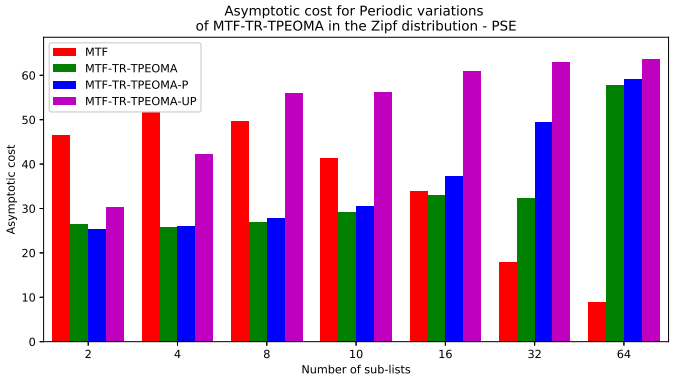


Figure 5.31: Asymptotic cost of Periodic variations of MTF-TR-TPEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

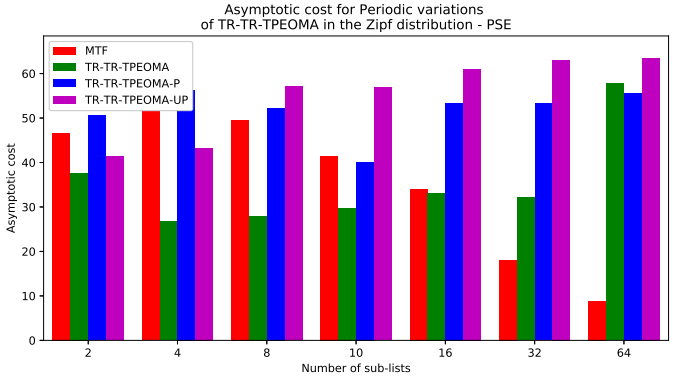


Figure 5.32: Asymptotic cost of Periodic variations of TR-TR-TPEOMA in the Zipf distribution. PSE with period $T = 30$ and $k = \{k : 2, 4, 8, 10, 16, 32, 64\}$.

further minimize the asymptotic and amortized costs in the PSE.

For the TPEOMA-augmented hierarchical schemes, the results in the case of MSEs were comparable to those obtained for the PEOMA-augmented variants. However, the PEOMA-augmented hierarchical schemes were overall, superior, to the TPEOMA. In the PSEs, the TPEOMA-enhanced schemes performed less than expected and in instances where the outer-list context was the TR, the MTF scheme had a superior asymptotic and amortized costs. When the "Periodic" and "UnPeriodic" variations were incorporated to the TPEOMA-enhanced schemes their performance further degraded. It is our conclusion that the TPEOMA-augmented hierarchical schemes are

not preferred for Periodic NSEs. In such Environments, the PEOMA-augmented hierarchical schemes yielded superior results to all the other schemes heretofore considered.

As mentioned in Section 4.10, 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. This is currently unresolved.

6

Summary and Conclusion

6.1 Summary of the Thesis

This thesis considered the problem of minimizing the asymptotic search cost of data structures in Non-stationary Environments (NSEs) by working with "Adaptive" Data Structures (ADS) that are based on the concept of sub-structure hierarchies. The goal of using such an adaptive data structure is to capture or learn the "unknown" dependent probabilistic distribution of the query Environment, and to thus adjust the corresponding elements to reflect the underlying dependence of the accesses.

This research improved on the work pioneered by Amer and Oommen in [3] by designing hierarchical Lists-on-List (LOL) data-structures that incorporate the state-of-the-art Object Migration Automaton (OMA) reinforcement schemes. The latter are based on the theory of Learning Automata (LA), and have been proposed so as to capture elements that have a probabilistic dependence of being jointly queried at

any particular time instant. By using the groups dictated by the OMA, we have been able to determine the elements which are to be within the same sublist context or "sub"-structure, so that they are moved "en masse" towards the head of the list with the associated re-organization rule.

The central idea motivating the hierarchical LOL "sub" structure involves the partitioning of a list of size n into k sub-lists where the reinforcement scheme groups dependent elements from the Environment within the same sub-list. The goal, of course, is so that the re-organization rule operates on the list sub-context "en masse" in minimizing the query access costs asymptotically. To achieve this, our work incorporated the previously-designed machines[1], the EOMA, the PEOMA and the TPEOMA into the LOL hierarchical schemes, and the results showed that the performances were superior to the original OMA-enhanced versions proposed in [3].

Chapter 2 introduced the theory of LA which forms the framework for the OMA used in learning the true partition of the objects into groups. The chapter first surveyed the fixed and variable structure stochastic automata, as well as the estimator and pursuit schemes that constituted the backbone of the PEOMA and TPEOMA algorithms. This chapter also surveyed other partitioning schemes such as the Basic Adaptive Method (BAM), the Stochastic Move Automaton (SMA) and the Modified Linear Reward Penalty Scheme (ML_{RP}). Chapter 2 also surveyed the fundamental adaptive list organizing schemes, i.e., the Move-to-Front (MTF) and the Transposition (TR) rules, which constitute the primitive rules used in the hierarchical LOL data-structures. The MTF and TR rules are also the baseline "de-facto" schemes in NSEs, which our EOMA, PEOMA, and TPEOMA-augmented approaches have used for empirical analysis in addition to the OMA-augmented schemes which earlier improved on the MTF and TR in [3].

Chapter 3 explained the rationale for using data "sub"-structures in designing the hierarchical schemes, and justified the concept of statically capturing the dependence. The chapter also introduced the MTF-MTF, MTF-TR, TR-MTF and TR-TR hierarchical schemes, and in the body of this chapter, we saw that in a NSE, the MTF and

[1]The EOMA, PEOMA and TPEOMA stand for the "Enhanced-Object Migration Automaton", "Pursuit Enhanced-Object Migration Automaton", and the "Transitivity Pursuit Enhanced-Object Migration Automaton" respectively.

TR stand-alone rules performed better than the vanilla hierarchical schemes, because in the latter, the sublist maintained a static ordering and did not reflect the probabilistic dependence of elements in the query Environment when moving elements "en masse" towards the head of the list.

In Chapter 4, the hierarchical schemes were augmented with the EOMA. In this chapter, we explained the significance of the EOMA over the OMA, inasmuch as the EOMA mitigates the deadlock scenario in the OMA. The performances of the EOMA-augmented hierarchical schemes in the Markovian Switching Environment (MSE) and the Periodic Switching Environment (PSE) were shown to be, in general, superior to the OMA-enhanced versions, and indeed, to the MTF and TR rules themselves. This was especially the case when the outer-list context of the hierarchical scheme was the MTF. When the outer-list context was the TR rule, the performances were at-par for Environments with L-shaped distributions like the Lotka and Exponential distributions.

Chapter 5 augmented the hierarchical schemes with the PEOMA and the TPEOMA reinforcement algorithms that incorporate the Estimators and consequently the Pursuit concept to better optimize the dependence-capturing mechanism of the system. In this chapter, we showed that the PEOMA-augmented hierarchical schemes are, in general, an order of magnitude superior to the EOMA-augmented schemes in the MSE, while for PSEs, the PEOMA-augmented schemes were superior to the EOMA variants.

The TPEOMA-augmented hierarchical schemes on the other hand, boasted an order of magnitude superior performance to the EOMA-augmented schemes in the MSE. However, in PSEs, the TPEOMA-augmented schemes were deemed to not be suitable for such NSEs because the action of incorporating the Transitivity relationship between query pairs to infer query dependence rendered the results to be unstable in such Environments.

In both Chapters 4 and 5, we implemented the Periodic variants for the LOLs. These variants consisted of using the "Periodic" and "UnPeriodic" additions to the hierarchical schemes, where the LOL data-structure was either provided with or not provided with the knowledge of the period from the Environment respectively, when

the query space changed. These hierarchical schemes that possessed the knowledge of the periods were superior to those that did not except for the TPEOMA-augmented schemes. In general, the "Periodic" or "UnPeriodic" enhancements were preferred in NSEs.

6.2 Conclusion of the Thesis

To summarize, the conclusive results of this research are as follows:

- The EOMA-augmented hierarchical schemes performed better than the original OMA-augmented schemes that used the novel idea of a hierarchical LOL approach;
- The EOMA-augmented hierarchical schemes were superior to the MTF and TR rules when the outer-list context was the MTF;
- The PEOMA-augmented hierarchical schemes were as order of magnitude superior to the EOMA-augmented schemes, as well as to the MTF and TR stand-alone rules in MSEs;
- The PEOMA-augmented hierarchical schemes were also superior to the EOMA-augmented schemes, and also superior to the stand-alone MTF and TR schemes in PSEs even when the outer-list context was the TR;
- The TPEOMA-augmented hierarchical schemes also achieved superior performances to the EOMA-augmented schemes, and to the MTF and TR stand-alone rules in MSEs;
- The TPEOMA-augmented hierarchical schemes were shown to be unsuitable for PSEs;
- The "Periodic" and "UnPeriodic" versions of the EOMA and PEOMA-augmented hierarchical schemes yielded a superior performance in PSEs to those without such additions.

In conclusion, the key contribution of this thesis is the enhancement of the hierarchical LOL approach for learning "locality of reference" in NSEs, with the EOMA, PEOMA and TPEOMA reinforcement paradigms. The resultant enhanced hierarchical schemes designed by this research are currently the state-of-the-art methods for adaptive singly-linked lists operating in NSEs.

6.3 Publications from the Thesis

The publications resulting from this research includes:

- Bisong, E. and Oommen, B.J., On Utilizing Enhanced Object Partitioning for Optimizing Self-Organizing Lists in Environments with Locality of Reference. (To be Submitted)
- Bisong, E. and Oommen, B.J., On Utilizing *Pursuit*-Enhanced Object Partitioning for Optimizing Self-Organizing Lists in Environments with Locality of Reference. (To be Submitted)
- Bisong, E. and Oommen, B.J., On Utilizing *Transitivity* and *Pursuit*-Enhanced Object Partitioning for Optimizing Self-Organizing Lists in Environments with Locality of Reference. (To be Submitted)

6.4 Future Work

The research advanced in this work opens the door to further extend and optimize the task of designing ADSs in terms of a hierarchy of primitive data "sub"-structures. As mentioned above, in our work, SLLs were optimized by utilizing hierarchical substructures that grouped similar elements together using the EOMA, PEOMA and TPEOMA. A future work will definitely be to implement these state-of-the-art object partitioning algorithms as the dependence capturing mechanisms for doubly-linked-lists on singly-linked-lists, singly-linked-lists on doubly-linked-list and for doubly-linked-lists on doubly-linked-lists. Each of these topics can lead to potential theses in their own rights.

Another extension to this work would be to package these solutions as a programming library for the development community. By doing this, the end-user would be able to observe how these algorithms worked in different real-life scenarios and use-cases. This will, certainly, be fundamental for a wider adoption of these data organizing mechanisms, and for their use in the computing and IT industries.

As mentioned in Section 4.10 and Section 5.16, an examination of the various graphs that we have obtained in Chapters 4 and 5 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 an avenue for much further research.

Finally, as the results presented so far have been empirical in nature, the rigorous formal analysis of these schemes is open. Clearly, such a theoretical endeavour will be far from trivial. Indeed, even the mathematical tools to analyze these LA machines have not been formalized.

Bibliography

[1] S. Albers and M. Mitzenmacher. Average case analyses of list update algorithms, with applications to data compression. *Algorithmica*, 21(3):312–329, 1998.

[2] S. Albers and J. Westbrook. Self-organizing data structures. In *Online algorithms*, pages 13–51. Springer, 1998.

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

[4] E. J. Anderson, P. Nash, and R. R. Weber. A counterexample to a conjecture on optimal list ordering. *Journal of Applied Probability*, 19(3):730–732, 1982.

[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] R. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press, Princeton, N.J., 1961.

[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] J. R. Bitner. Heuristics that dynamically organize data structures. *SIAM Journal on Computing*, 8(1):82–110, 1979.

[9] R. R. Bush and F. Mosteller. Stochastic models for learning. 1955.

[10] P. Chassaing. Optimality of move-to-front for self-organizing data structures with locality of references. *The Annals of Applied Probability*, pages 1219–1240, 1993.

[11] F. R. K. Chung, D. J. Hajela, and P. D. Seymour. Self-organizing sequential search and hilbert's inequalities. *Journal of Computer and System Sciences*, 36(2):148–157, 1988.

[12] R. Cogburn. The ergodic theory of Markov chains in random environments. *Z. Wahrsch. Verw. Gebiete*, 66(1):109–128, 1984.

[13] J. Dong. Time reversible self-organizing sequential search algorithms. 1998.

[14] 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.

[15] 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.

[16] K. Fu. Learning control systems–review and outlook. *IEEE transactions on Automatic Control*, 15(2):210–221, 1970.

[17] K. Fu and R. W. McLaren. *An Application of Stochastic Automata to the Synthesis of Learning Systems*. TR-EE. School of Electrical Engineering, Purdue University, 1965.

[18] K. Fu and G. J. McMurtry. *A study of stochastic automata as models of adaptive and learning controllers*. Purdue University School of Electrical Engineering, 1966.

[19] K. Fu and Z. J. Nikolic. On some reinforcement schemes and their relation to stochastic approximation. *IEEE transactions on Automatic Control*, pages 756–58, 1966.

[20] K. Fu and M. D. Waltz. A computer simulated learning control system. *IEEE International Convention Record*, pages 190–201, 1966.

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

[22] G. H. Gonnet, J. I. Munro, and H. Suwanda. Exegesis of self-organizing linear search. *SIAM Journal on Computing*, 10(3):613–637, 1981.

[23] W. J. Hendricks. An account of self-organizing systems. *SIAM Journal on Computing*, 5(4):715–723, 1976.

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

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

[26] J. Iacono. Alternatives to splay trees with o (log n) worst-case access times. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 516–522. Society for Industrial and Applied Mathematics, 2001.

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

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

[29] Y. C. Kan and S. M. Ross. Optimal list order under partial memory constraints. *Journal of Applied Probability*, 17(4):1004–1015, 1980.

[30] D. E. Knuth. *The art of computer programming*, volume 3. Pearson Education, 1997.

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

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

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

[34] D. C. Y. Ma. Object partitioning by using learning automata. 1987.

[35] 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.

[36] J. McCabe. On serial files with relocatable records. *Operations Research*, 13(4):609–618, 1965.

[37] K. S. Narendra and M. A. L. Thathachar. *Learning automata: an introduction*. Courier Corporation, 2012.

[38] M. F. Norman. On the linear model with two absorbing barriers. *Journal of Mathematical Psychology*, 5(2):225–241, 1968.

[39] M. F. Norman. Some convergence theorems for stochastic learning models with distance diminishing operators. *Journal of Mathematical Psychology*, 5(1):61–101, 1968.

[40] M. F. Norman. *Markov processes and learning models*, volume 84. Academic Press New York, 1972.

[41] B. J. Oommen and J. Dong. Generalized swap-with-parent schemes for self-organizing sequential linear lists. In *International Symposium on Algorithms and Computation*, pages 414–423. Springer, 1997.

[42] 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.

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

[44] B. J. Oommen and E. R. Hansen. List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations. *SIAM Journal on Computing*, 16(4):705–716, 1987.

[45] B. J. Oommen, E. R. Hansen, and J. I. Munro. Deterministic optimal and expedient move-to-rear list organizing strategies. *Theoretical Computer Science*, 74(2):183–197, 1990.

[46] B. J. Oommen and D. C. Y. Ma. *Stochastic automata solutions to the object partitioning problem*. Carleton University, School of Computer Science, 1986.

[47] B. J. Oommen and D. C. Y. Ma. Deterministic learning automata solutions to the equipartitioning problem. *IEEE Transactions on Computers*, 37(1):2–13, 1988.

[48] B. J. Oommen and D. T. H. Ng. An optimal absorbing list organization strategy with constant memory requirements. *Theoretical computer science*, 119(2):355–361, 1993.

[49] B. J. Oommen and J. Zgierski. A learning automaton solution to breaking substitution ciphers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:185–192, 1993.

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

[51] S. M. Ross. *Introduction to probability models*. Academic press, 2014.

[52] F. Schulz. Two new families of list update algorithms. In *International Symposium on Algorithms and Computation*, pages 100–109. Springer, 1998.

[53] O. G. Selfridge. Tracking and trailing: adaptation in movement strategies. *Unpublished draft*, 1978.

[54] I. J. Shapiro and K. S. Narendra. Use of stochastic automata for parameter self-optimization with multimodal performance criteria. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):352–360, 1969.

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

[56] A. Shirvani and B. J. Oommen. The advantages of invoking transitivity in enhancing pursuit-oriented object migration automata. 2017.

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

[58] A. Tenenbaum. Simulations of dynamic sequential search algorithms. *Communications of the ACM*, 21(9):790–791, 1978.

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

[60] M. A. L. Thathachar and P. S. Sastry. Estimator algorithms for learning automata. 1986.

[61] M. L. Tsetlin. On behaviour of finite automata in random medium. *Avtom I Telemekhanika*, 22:1345–1354, 1961.

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

[63] V. I. Varshavskii, M. V. Meleshina, and M. L. Tsetlin. Behavior of automata in periodic random media and the problem of synchronization at presence of noise. *Problemy Peredachi Informatsii*, 1(1):65–71, 1965.

[64] V. I. Varshavskii and I. P. Vorontsova. On the behavior of stochastic automata with a variable structure. *Avtomatika i Telemekhanika*, 24(3):353–360, 1963.

[65] I. P. Vorontsova. Algorithms for changing stochastic automata transition probabilities. *Problemy Peredachi Informatsii*, 1(3):122–126, 1965.

[66] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[67] 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.

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

[69] C. T. Yu, C. Suen, K. Lam, and M. K. Siu. Adaptive record clustering. *ACM Transactions on Database Systems (TODS)*, 10(2):180–204, 1985.

Appendices

A

OMA-Augmented Hierarchical SLLs Results

This Appendix contains the experimental results for the OMA Hierarchical SLL on SLLs. The results contrast the performance of the MTF and TR with the hierarchical MTF-MTF-OMA, TR-MTF-OMA, MTF-TR-OMA, and TR-TR-OMA 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-OMA	35.53	36.34	37.38	38.18	28.33	28.99	9.42	10.12	39.89	40.95
MTF-TR-OMA	35.45	36.53	37.20	38.01	28.12	29.19	11.26	11.66	39.70	40.80
TR-MTF-OMA	32.27	33.36	34.02	35.29	26.09	27.25	11.78	12.17	36.27	37.69
TR-TR-OMA	32.26	33.47	33.92	35.80	26.05	27.20	11.96	13.41	36.34	37.74
$k = 4$	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-OMA	26.46	27.83	26.87	28.12	24.67	26.55	10.18	10.93	27.07	28.61
MTF-TR-OMA	29.69	30.95	29.85	30.86	27.78	29.55	12.34	12.85	30.33	31.83
TR-MTF-OMA	25.31	26.45	26.24	27.11	23.75	25.55	10.52	10.95	26.23	27.88
TR-TR-OMA	29.69	30.87	29.11	30.09	26.97	29.12	13.41	13.99	28.96	30.61
$k = 8$	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-OMA	20.27	23.18	20.24	23.17	20.14	22.36	12.04	12.72	21.08	24.52
MTF-TR-OMA	28.80	31.85	28.74	32.67	29.22	32.07	14.39	15.12	29.41	33.85
TR-MTF-OMA	19.95	23.06	19.90	23.53	20.17	22.75	11.94	13.00	21.00	24.40
TR-TR-OMA	28.53	32.13	28.68	31.99	28.24	30.99	14.91	15.79	29.05	32.45
$k = 16$	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-OMA	16.37	20.70	16.43	20.88	16.17	20.28	16.36	17.09	17.14	21.20
MTF-TR-OMA	34.25	38.50	33.86	38.29	34.21	38.68	27.77	27.38	34.18	38.80
TR-MTF-OMA	16.19	20.41	16.16	20.81	16.28	20.19	16.13	17.15	16.59	20.86
TR-TR-OMA	34.90	39.49	34.08	38.81	35.06	39.18	27.15	27.67	34.79	38.91
$k = 32$	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-OMA	13.62	15.55	13.65	15.46	13.64	15.39	15.57	18.36	13.71	15.40
MTF-TR-OMA	41.39	43.37	41.52	43.27	41.79	43.35	40.64	41.18	41.53	43.27
TR-MTF-OMA	13.68	15.46	13.56	15.31	13.64	15.49	16.07	18.97	13.74	15.69
TR-TR-OMA	41.96	44.51	41.71	44.55	42.01	44.44	38.81	40.39	42.34	45.21
$k = 64$	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-OMA	12.63	12.65	12.64	12.64	12.68	12.67	12.51	12.63	12.64	12.63
MTF-TR-OMA	50.11	50.13	50.12	50.00	49.84	50.01	50.08	50.00	50.14	50.04
TR-MTF-OMA	12.66	12.67	12.66	12.69	12.52	12.61	12.61	12.70	12.69	12.69
TR-TR-OMA	50.05	49.96	49.89	50.00	49.82	50.13	50.20	50.18	49.86	50.00

Table A.1: OMA-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-OMA	53.42	53.39	56.27	56.16	33.28	34.73	8.42	9.53	60.09	60.09
MTF-TR-OMA	52.96	52.81	56.34	56.18	36.14	34.45	8.82	9.67	60.01	60.12
TR-MTF-OMA	51.69	52.28	55.31	55.54	32.69	33.39	7.13	9.02	59.00	59.09
TR-TR-OMA	47.61	47.88	50.49	50.83	26.84	27.46	11.26	12.36	56.04	56.27
$k = 4$	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-OMA	63.34	63.32	63.63	63.58	58.75	58.69	14.40	15.03	63.40	63.43
MTF-TR-OMA	63.09	63.02	63.36	63.31	57.93	57.91	14.03	14.92	63.26	63.27
TR-MTF-OMA	64.30	64.36	64.39	64.33	62.20	62.20	15.28	17.21	64.19	64.12
TR-TR-OMA	63.97	63.99	64.00	64.08	61.46	61.42	19.81	20.37	64.06	64.06
$k = 8$	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-OMA	63.16	63.15	63.31	63.28	61.53	61.57	41.54	41.83	63.29	63.31
MTF-TR-OMA	63.23	63.25	63.31	63.35	62.14	62.12	41.42	41.35	63.27	63.31
TR-MTF-OMA	63.23	63.20	63.44	63.34	61.96	61.89	44.30	44.90	63.43	63.41
TR-TR-OMA	63.33	63.32	63.42	63.34	62.50	62.51	27.89	29.93	63.38	63.38
$k = 16$	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-OMA	63.21	63.16	63.24	63.24	62.67	62.63	52.53	52.38	63.20	63.19
MTF-TR-OMA	63.47	63.44	63.49	63.48	63.27	63.28	56.93	57.07	63.21	63.24
TR-MTF-OMA	63.16	63.17	63.19	63.20	62.60	62.67	52.91	52.83	63.20	63.18
TR-TR-OMA	63.48	63.46	63.53	63.51	63.28	63.34	57.69	57.66	63.31	63.28
$k = 32$	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-OMA	63.04	63.06	63.02	63.08	62.88	62.87	58.60	58.59	62.91	62.94
MTF-TR-OMA	63.43	63.46	63.49	63.45	63.49	63.48	62.78	62.84	63.19	63.21
TR-MTF-OMA	63.12	63.05	63.03	63.07	62.81	62.83	58.53	58.54	62.92	62.96
TR-TR-OMA	63.46	63.46	63.51	63.48	63.47	63.46	62.91	62.88	63.30	63.23
$k = 64$	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-OMA	62.49	62.47	62.42	62.45	62.36	62.38	61.12	61.09	62.25	62.21
MTF-TR-OMA	63.41	63.41	63.48	63.47	63.45	63.42	63.31	63.33	63.40	63.43
TR-MTF-OMA	62.33	62.44	62.45	62.41	62.34	62.36	61.04	61.07	62.26	62.22
TR-TR-OMA	63.48	63.45	63.34	63.44	63.46	63.41	63.36	63.40	63.39	63.46

Table A.2: OMA-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-OMA	29.23	31.43	30.96	32.88	23.84	25.62	6.38	6.97	33.46	35.24
MTF-TR-OMA	29.20	31.21	30.79	32.90	23.92	25.33	6.41	6.86	33.32	35.44
TR-MTF-OMA	26.00	28.75	27.26	30.07	22.00	24.00	7.08	7.62	30.11	32.57
TR-TR-OMA	28.26	31.95	27.56	30.18	24.68	25.98	8.35	8.71	30.10	32.15
MTF-MTF-OMA-P	25.02	27.97	26.68	29.32	21.10	23.50	4.62	5.12	29.37	32.13
MTF-MTF-OMA-UP	28.27	31.23	29.55	32.62	23.88	25.81	5.22	6.18	32.34	35.13
MTF-TR-OMA-P	25.33	27.84	26.36	29.39	20.46	22.66	4.47	4.88	29.39	32.00
MTF-TR-OMA-UP	27.78	31.06	29.22	32.37	23.96	25.91	8.86	9.98	32.84	35.66
TR-MTF-OMA-P	21.66	24.88	23.20	25.76	17.96	21.11	6.63	7.33	26.03	29.10
TR-MTF-OMA-UP	24.78	28.25	25.58	29.42	22.01	24.01	8.59	9.28	28.89	32.36
TR-TR-OMA-P	22.16	26.22	23.35	26.78	20.09	22.80	5.09	6.90	25.98	28.93
TR-TR-OMA-UP	25.12	28.36	25.97	29.13	21.73	23.90	5.55	6.17	29.18	32.37
$k = 4$	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-OMA	21.40	22.87	21.89	23.71	20.61	22.76	7.73	8.52	22.70	24.83
MTF-TR-OMA	24.79	26.36	25.28	26.85	24.71	26.84	9.29	10.26	25.78	27.59
TR-MTF-OMA	20.74	22.38	21.42	23.39	20.26	23.08	8.96	9.85	21.88	23.98
TR-TR-OMA	24.13	26.03	24.37	26.32	24.87	26.85	9.73	10.33	24.93	27.22
MTF-MTF-OMA-P	14.86	17.05	15.46	17.55	14.43	16.73	6.67	7.77	15.96	18.06
MTF-MTF-OMA-UP	19.25	24.47	20.24	26.30	22.22	27.35	13.09	14.75	20.41	25.41
MTF-TR-OMA-P	15.47	17.41	15.98	17.43	14.66	16.43	7.82	8.89	16.34	19.27
MTF-TR-OMA-UP	21.23	25.00	22.94	26.35	25.06	29.50	12.38	15.10	20.93	26.05
TR-MTF-OMA-P	14.79	16.86	15.08	16.96	13.04	15.78	6.99	8.07	14.97	17.31
TR-MTF-OMA-UP	19.69	24.64	20.61	24.63	21.58	26.59	11.43	13.59	20.94	25.66
TR-TR-OMA-P	14.33	16.70	14.97	17.34	13.92	16.54	10.26	11.98	15.14	17.47
TR-TR-OMA-UP	19.08	24.70	19.95	24.87	20.56	25.09	11.43	15.18	20.02	24.88
$k = 8$	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-OMA	18.25	21.01	18.30	20.62	18.76	20.83	9.63	10.37	18.40	21.20
MTF-TR-OMA	30.27	33.27	31.35	34.15	31.60	34.41	11.93	12.70	30.31	33.99
TR-MTF-OMA	19.08	21.05	18.45	20.81	18.43	20.18	9.78	10.67	18.46	21.41
TR-TR-OMA	31.78	34.40	31.11	35.14	31.00	33.92	13.69	14.55	30.95	34.35
MTF-MTF-OMA-P	11.38	14.35	10.96	14.67	11.58	13.69	8.01	8.97	11.53	15.12
MTF-MTF-OMA-UP	24.86	30.93	26.80	33.04	28.59	33.95	17.43	21.43	24.99	34.39
MTF-TR-OMA-P	11.91	15.00	11.03	15.05	10.74	13.22	11.93	13.46	11.52	15.28
MTF-TR-OMA-UP	28.91	36.50	30.47	37.71	33.46	40.04	24.06	27.79	31.98	39.36
TR-MTF-OMA-P	11.60	14.16	11.23	14.15	12.20	14.20	8.40	10.28	12.84	15.77
TR-MTF-OMA-UP	26.91	33.51	24.90	32.80	25.70	32.42	18.59	21.71	25.64	32.88
TR-TR-OMA-P	12.28	15.48	11.18	15.23	12.00	14.68	14.55	16.25	11.64	16.46
TR-TR-OMA-UP	26.03	31.58	25.16	32.60	27.86	33.72	18.13	22.19	24.44	31.75

Table A.3: OMA-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-OMA	17.16	19.07	16.52	18.86	16.05	18.03	13.25	14.24	16.95	19.09
MTF-TR-OMA	48.03	51.33	47.97	51.73	47.76	50.49	26.94	26.72	48.46	52.09
TR-MTF-OMA	15.80	18.18	15.31	18.23	16.31	18.12	13.16	14.17	16.21	18.36
TR-TR-OMA	52.97	55.02	52.52	54.65	50.43	53.09	28.12	28.00	52.23	53.77
MTF-MTF-OMA-P	9.42	12.28	9.88	12.77	10.20	12.75	8.79	10.07	10.44	12.84
MTF-MTF-OMA-UP	34.47	41.18	33.53	42.53	32.86	41.48	26.29	30.06	29.14	38.31
MTF-TR-OMA-P	12.21	15.91	11.85	15.52	11.72	15.14	13.70	16.42	12.19	15.88
MTF-TR-OMA-UP	37.06	43.84	36.88	44.00	36.12	43.41	39.60	42.56	37.91	44.77
TR-MTF-OMA-P	10.43	13.03	9.93	12.68	10.15	12.71	8.49	9.89	9.97	12.39
TR-MTF-OMA-UP	33.35	42.18	30.27	40.41	31.64	40.08	28.18	31.58	31.41	41.83
TR-TR-OMA-P	15.46	19.14	14.40	18.66	17.48	20.01	20.02	22.21	15.72	19.00
TR-TR-OMA-UP	32.42	41.28	34.10	43.56	31.99	40.53	25.07	30.00	32.56	41.33
$k = 32$	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-OMA	10.24	11.48	10.14	11.57	10.23	11.72	13.09	14.76	10.44	11.60
MTF-TR-OMA	66.34	64.99	66.26	64.90	66.14	64.56	50.57	48.39	66.06	64.11
TR-MTF-OMA	10.16	11.58	10.17	11.49	10.22	11.58	13.15	14.75	10.52	11.91
TR-TR-OMA	65.44	62.91	65.88	63.44	64.14	61.59	47.13	48.17	64.01	62.13
MTF-MTF-OMA-P	2.15	3.75	2.30	4.25	1.83	3.64	6.13	8.11	2.95	4.79
MTF-MTF-OMA-UP	6.85	16.59	4.91	16.17	6.20	17.41	28.02	34.70	12.59	21.71
MTF-TR-OMA-P	2.52	6.88	2.37	7.22	2.33	7.19	14.75	19.41	3.52	8.03
MTF-TR-OMA-UP	17.19	27.32	19.97	28.93	15.78	27.57	41.68	46.22	24.26	32.92
TR-MTF-OMA-P	2.36	4.30	2.29	4.03	2.37	4.20	6.20	8.27	2.73	4.86
TR-MTF-OMA-UP	7.74	17.35	8.05	18.48	6.44	17.56	26.84	35.54	10.53	21.42
TR-TR-OMA-P	4.58	11.31	3.99	10.81	4.99	12.70	25.92	28.14	6.31	13.31
TR-TR-OMA-UP	6.86	16.77	8.95	19.16	9.27	17.00	26.50	34.47	11.93	22.68
$k = 64$	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-OMA	8.89	8.91	8.89	8.91	8.89	8.91	8.84	8.91	8.89	8.90
MTF-TR-OMA	64.99	64.95	64.96	64.93	64.92	64.89	64.84	64.75	64.91	64.85
TR-MTF-OMA	8.89	8.91	8.89	8.91	8.89	8.91	8.84	8.90	8.89	8.90
TR-TR-OMA	64.92	64.88	64.87	64.84	64.92	64.88	64.74	64.29	64.93	64.84
MTF-MTF-OMA-P	0.49	0.55	0.49	0.54	0.49	0.54	0.44	0.57	0.49	0.54
MTF-MTF-OMA-UP	0.56	0.85	0.56	1.00	0.56	0.94	0.51	1.30	0.55	1.02
MTF-TR-OMA-P	4.32	4.67	7.47	7.78	5.34	5.71	4.07	4.98	4.79	5.12
MTF-TR-OMA-UP	3.13	4.03	0.56	2.32	2.36	4.09	0.79	4.65	3.53	4.89
TR-MTF-OMA-P	0.49	0.55	0.50	0.55	0.49	0.56	0.44	0.56	0.49	0.55
TR-MTF-OMA-UP	0.56	0.91	0.56	0.93	0.56	0.56	0.94	0.51	0.55	0.89
TR-TR-OMA-P	5.43	6.13	5.46	5.99	5.06	5.54	2.59	4.75	4.76	5.32
TR-TR-OMA-UP	0.56	0.90	0.56	0.90	0.56	0.91	0.51	1.48	0.55	1.02

Table A.4: OMA-Augmented Hierarchical SLLs Results for PSE with $T = 30$ and $k = 16, 32, 64$.

B

EOMA-Augmented Hierarchical SLLs Results

This Appendix contains the experimental results for the OMA Hierarchical SLL 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 B.1: 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 B.2: 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 B.3: 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 B.4: EOMA-Augmented Hierarchical SLLs Results for PSE with $T = 30$ and $k = 16, 32, 64$.

C

PEOMA-Augmented Hierarchical SLLs Results

This Appendix contains the experimental results for the OMA Hierarchical SLL on SLLs. The results contrast the performance of the MTF and TR with the hierarchical MTF-MTF-PEOMA, TR-MTF-PEOMA, MTF-TR-PEOMA, and TR-TR-PEOMA 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-PEOMA	7.66	10.47	10.89	14.36	1.25	2.61	2.33	2.37	21.56	23.47
MTF-TR-PEOMA	7.71	10.56	10.83	14.24	1.24	2.60	3.24	3.27	21.59	23.51
TR-MTF-PEOMA	5.83	8.63	8.21	11.41	0.98	3.42	2.18	2.26	18.60	20.34
TR-TR-PEOMA	5.84	8.62	8.18	11.41	0.97	3.00	2.19	2.27	18.59	20.37
$k = 4$	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-PEOMA	8.61	10.22	10.76	12.11	1.25	2.38	4.75	5.13	12.45	13.51
MTF-TR-PEOMA	8.64	10.45	10.79	12.40	1.25	2.37	4.74	5.32	12.12	13.51
TR-MTF-PEOMA	6.48	8.28	8.39	9.88	0.99	2.20	5.05	5.74	10.04	11.38
TR-TR-PEOMA	6.48	8.44	8.37	10.12	0.96	2.34	4.82	6.60	10.03	11.52
$k = 8$	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-PEOMA	5.80	6.97	6.73	7.77	1.25	2.32	2.45	4.00	6.76	7.79
MTF-TR-PEOMA	5.35	7.14	6.21	7.87	1.25	2.63	2.97	4.23	6.77	8.31
TR-MTF-PEOMA	4.67	5.95	6.00	7.13	0.96	2.04	2.88	4.65	5.61	6.71
TR-TR-PEOMA	5.07	6.84	6.49	8.05	0.98	2.29	2.99	4.77	6.34	7.91
$k = 16$	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-PEOMA	3.49	4.59	3.48	4.51	1.68	2.95	1.95	4.09	3.20	4.28
MTF-TR-PEOMA	3.41	5.34	3.49	5.38	3.01	5.02	1.56	4.60	3.38	5.27
TR-MTF-PEOMA	3.36	4.38	3.48	4.51	2.65	3.80	1.42	3.92	3.11	4.18
TR-TR-PEOMA	3.47	5.39	3.38	5.31	1.46	3.72	1.49	4.85	2.86	4.72
$k = 32$	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-PEOMA	1.73	2.99	1.50	2.75	1.49	2.75	1.06	3.00	1.24	2.34
MTF-TR-PEOMA	1.61	3.31	4.17	5.83	1.42	3.19	1.15	3.45	1.49	3.21
TR-MTF-PEOMA	1.62	2.90	1.59	2.79	3.79	5.29	1.14	3.13	1.35	2.59
TR-TR-PEOMA	1.49	3.16	1.47	3.23	4.42	6.33	3.18	5.06	1.29	3.06
$k = 64$	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-PEOMA	0.49	0.92	0.50	0.92	0.49	0.87	0.44	0.87	0.39	0.77
MTF-TR-PEOMA	0.49	0.92	0.50	0.93	0.50	0.92	0.44	0.87	0.24	0.60
TR-MTF-PEOMA	0.49	0.90	0.49	0.92	0.49	0.91	0.44	0.81	0.35	0.68
TR-TR-PEOMA	0.50	0.91	0.50	0.90	0.50	0.95	0.44	0.88	0.29	0.66

Table C.1: PEOMA-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-PEOMA	63.02	63.51	64.16	65.00	37.63	41.34	3.28	3.31	66.14	65.05
MTF-TR-PEOMA	62.43	63.44	64.58	64.75	38.66	42.27	3.28	3.31	64.46	64.34
TR-MTF-PEOMA	73.21	72.98	73.43	72.45	62.55	65.24	2.55	2.61	69.50	69.01
TR-TR-PEOMA	73.48	72.74	72.33	72.07	62.81	65.31	2.54	2.61	69.69	68.99
$k = 4$	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-PEOMA	57.98	59.54	59.84	61.08	44.44	46.96	16.72	18.69	61.65	62.27
MTF-TR-PEOMA	57.24	59.28	60.24	60.99	43.73	46.94	19.49	20.19	61.19	62.12
TR-MTF-PEOMA	60.84	61.90	62.44	62.88	49.45	52.27	29.11	30.63	63.15	63.55
TR-TR-PEOMA	60.81	61.93	62.28	63.10	49.42	51.71	29.92	31.07	63.12	63.62
$k = 8$	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-PEOMA	59.23	60.64	60.36	61.31	52.18	54.45	22.32	25.84	60.85	61.89
MTF-TR-PEOMA	59.48	60.73	60.44	61.53	52.55	54.49	22.57	26.05	60.60	61.72
TR-MTF-PEOMA	59.83	60.95	60.75	61.61	53.50	55.25	23.82	26.73	61.09	62.05
TR-TR-PEOMA	59.68	60.93	60.94	61.72	53.35	55.20	23.41	26.89	60.90	61.93
$k = 16$	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-PEOMA	58.92	61.10	59.44	61.39	52.57	56.42	32.25	35.23	57.60	60.62
MTF-TR-PEOMA	59.55	61.73	60.87	62.29	52.98	57.31	32.29	36.04	58.41	61.35
TR-MTF-PEOMA	59.06	61.14	60.03	61.55	51.93	56.45	32.26	35.67	58.10	60.82
TR-TR-PEOMA	60.02	61.97	60.84	62.38	53.03	57.29	32.85	36.50	58.67	61.39
$k = 32$	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-PEOMA	56.45	59.66	56.78	59.78	51.42	56.90	46.02	48.13	51.45	57.79
MTF-TR-PEOMA	59.77	61.97	60.21	62.16	55.62	60.03	55.47	56.87	54.21	60.12
TR-MTF-PEOMA	56.29	59.60	56.82	59.81	51.82	57.15	46.01	48.15	50.86	57.53
TR-TR-PEOMA	59.82	62.01	60.50	62.30	55.37	60.05	55.44	56.88	54.39	60.16
$k = 64$	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-PEOMA	47.49	53.93	47.56	54.01	45.64	53.30	50.73	52.96	11.95	44.96
MTF-TR-PEOMA	59.60	61.83	59.40	61.86	58.12	61.43	61.52	62.06	16.93	50.52
TR-MTF-PEOMA	47.63	53.88	47.71	54.03	45.43	53.26	50.45	50.84	15.95	45.36
TR-TR-PEOMA	59.46	61.83	59.72	61.91	58.11	61.44	61.58	62.01	13.79	51.44

Table C.2: PEOMA-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-PEOMA	21.23	22.39	22.46	24.23	16.76	17.44	2.16	2.18	29.66	30.58
MTF-TR-PEOMA	21.36	22.35	25.19	26.73	16.86	17.32	2.15	2.19	29.57	30.59
TR-MTF-PEOMA	18.93	19.99	20.06	21.57	17.62	18.68	2.52	2.59	26.31	27.33
TR-TR-PEOMA	19.12	20.00	20.16	21.58	16.00	16.70	2.19	2.26	26.32	27.32
MTF-MTF-PEOMA-P	15.80	17.05	17.06	18.75	11.64	12.32	4.05	4.08	23.96	25.11
MTF-MTF-PEOMA-UP	18.60	19.82	19.79	21.49	14.27	14.95	2.17	2.20	26.74	27.83
MTF-TR-PEOMA-P	15.77	17.01	16.99	18.76	13.45	13.93	4.34	4.37	23.98	25.12
MTF-TR-PEOMA-UP	18.55	19.81	19.89	21.52	14.15	14.81	2.17	2.19	26.82	27.89
TR-MTF-PEOMA-P	13.46	14.70	14.55	16.03	10.31	11.81	4.04	4.11	20.74	21.87
TR-MTF-PEOMA-UP	61.43	17.48	17.35	18.86	13.50	14.84	2.19	2.26	23.53	24.60
TR-TR-PEOMA-P	13.48	14.59	14.69	16.04	9.45	11.04	3.72	3.79	20.77	21.83
TR-TR-PEOMA-UP	16.22	17.45	17.41	18.94	12.03	13.47	2.18	2.26	23.59	24.60
$k = 4$	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-PEOMA	20.78	20.83	21.41	21.41	18.42	18.93	3.51	3.83	21.82	21.85
MTF-TR-PEOMA	24.51	24.51	30.56	30.40	22.09	22.55	3.36	3.86	25.58	25.49
TR-MTF-PEOMA	19.59	19.73	20.21	20.27	17.18	17.85	3.86	4.31	20.75	20.79
TR-TR-PEOMA	23.30	23.34	23.92	23.90	20.82	21.24	3.51	3.73	24.44	24.43
MTF-MTF-PEOMA-P	13.25	13.50	13.91	14.07	11.10	11.53	6.24	6.67	14.30	14.48
MTF-MTF-PEOMA-UP	14.59	14.79	15.18	15.33	12.34	12.95	5.09	5.48	15.56	15.73
MTF-TR-PEOMA-P	13.31	13.53	13.93	14.11	11.09	11.94	8.39	8.83	14.28	14.45
MTF-TR-PEOMA-UP	14.55	14.79	15.19	15.35	12.34	12.85	5.77	6.05	15.52	15.72
TR-MTF-PEOMA-P	12.12	12.38	12.71	12.92	9.83	10.49	6.45	7.22	13.18	13.43
TR-MTF-PEOMA-UP	13.37	13.68	14.00	14.21	11.07	11.68	5.22	5.84	14.44	14.66
TR-TR-PEOMA-P	12.09	12.38	12.72	12.92	9.85	10.43	8.57	9.05	13.16	13.40
TR-TR-PEOMA-UP	13.35	13.66	13.98	14.23	11.07	11.98	5.97	6.54	14.44	14.66
$k = 8$	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-PEOMA	15.19	15.27	15.29	15.35	14.57	14.81	4.10	4.88	15.30	15.37
MTF-TR-PEOMA	27.30	27.19	37.22	37.08	26.64	26.64	5.56	5.90	27.42	27.33
TR-MTF-PEOMA	14.90	15.00	15.04	15.13	14.21	14.41	4.32	5.25	15.04	15.13
TR-TR-PEOMA	27.00	26.94	27.11	27.05	26.28	26.32	5.56	6.31	27.17	27.12
MTF-MTF-PEOMA-P	7.13	7.27	7.21	7.34	6.53	6.75	6.10	7.07	7.21	7.36
MTF-MTF-PEOMA-UP	7.71	7.87	7.78	7.95	7.11	7.41	5.42	6.90	7.79	7.99
MTF-TR-PEOMA-P	7.12	7.27	7.21	7.37	6.52	6.81	13.68	14.53	7.21	7.37
MTF-TR-PEOMA-UP	7.70	7.91	7.78	7.99	7.11	7.53	7.19	8.49	7.79	8.01
TR-MTF-PEOMA-P	6.85	7.00	6.94	7.12	6.16	6.38	6.50	7.48	6.95	7.11
TR-MTF-PEOMA-UP	7.43	7.59	7.54	7.22	6.73	7.11	6.75	8.01	7.54	7.74
TR-TR-PEOMA-P	6.85	7.03	6.96	7.12	6.16	6.42	12.99	14.06	6.98	7.12
TR-TR-PEOMA-UP	7.41	7.64	7.52	7.74	6.73	7.13	7.05	8.62	7.54	7.78

Table C.3: PEOMA-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-PEOMA	11.67	11.87	11.68	11.88	11.53	11.76	11.39	12.41	11.66	11.87
MTF-TR-PEOMA	40.48	40.44	58.43	58.33	40.33	40.25	25.92	24.25	40.44	40.41
TR-MTF-PEOMA	11.62	11.79	11.65	11.83	11.46	11.72	11.63	12.85	11.61	11.85
TR-TR-PEOMA	40.43	40.39	40.48	40.43	40.27	40.19	27.08	26.04	40.42	40.38
MTF-MTF-PEOMA-P	3.42	3.63	3.44	3.65	3.31	3.53	7.60	8.11	3.42	3.62
MTF-MTF-PEOMA-UP	3.71	4.00	3.71	4.01	3.58	4.02	20.01	20.19	3.69	3.98
MTF-TR-PEOMA-P	3.43	3.72	3.45	3.74	3.31	3.71	15.02	20.30	3.42	3.71
MTF-TR-PEOMA-UP	3.70	4.12	3.72	4.09	3.58	4.04	24.29	26.84	3.70	4.07
TR-MTF-PEOMA-P	3.38	3.63	3.40	3.62	3.24	3.49	6.93	8.90	3.38	3.60
TR-MTF-PEOMA-UP	3.66	3.94	3.68	3.96	3.51	3.91	17.89	20.19	3.66	3.93
TR-TR-PEOMA-P	3.38	3.72	3.41	3.68	3.24	3.74	16.41	20.80	3.38	3.76
TR-TR-PEOMA-UP	3.66	4.08	3.68	4.06	3.51	4.06	25.05	26.69	3.66	4.12
$k = 32$	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-PEOMA	9.79	11.03	9.79	10.88	9.76	11.07	16.99	17.39	9.77	11.00
MTF-TR-PEOMA	66.34	61.94	65.17	63.12	66.31	62.16	19.78	19.77	66.35	62.44
TR-MTF-PEOMA	9.78	11.09	9.78	11.00	9.76	11.12	16.89	17.41	9.77	10.86
TR-TR-PEOMA	66.34	62.46	66.34	62.18	66.35	62.09	19.51	19.97	66.34	62.26
MTF-MTF-PEOMA-P	1.49	2.76	1.49	2.87	1.54	3.05	16.23	16.93	1.47	2.84
MTF-MTF-PEOMA-UP	1.62	4.02	1.62	3.64	1.59	4.14	21.71	21.14	1.60	3.85
MTF-TR-PEOMA-P	1.48	6.49	1.49	6.44	1.46	6.73	46.35	47.28	1.48	6.06
MTF-TR-PEOMA-UP	1.62	6.37	1.62	6.31	1.60	6.41	29.33	29.20	1.61	6.32
TR-MTF-PEOMA-P	1.48	2.77	1.48	2.78	1.46	2.91	15.90	16.58	1.47	2.87
TR-MTF-PEOMA-UP	1.62	4.08	1.62	4.11	1.59	4.43	21.15	21.14	1.60	3.99
TR-TR-PEOMA-P	1.48	6.32	1.48	6.16	1.46	7.34	47.78	47.68	1.47	6.01
TR-TR-PEOMA-UP	1.62	6.09	1.62	5.22	1.59	6.06	30.28	29.28	1.60	5.62
$k = 64$	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-PEOMA	4.72	4.73	4.72	4.73	4.72	4.72	4.67	4.67	4.72	4.72
MTF-TR-PEOMA	4.72	4.73	6.85	6.85	4.72	4.72	4.67	4.67	4.72	4.72
TR-MTF-PEOMA	4.72	4.73	4.72	4.73	4.72	4.73	4.67	4.67	4.72	4.72
TR-TR-PEOMA	4.72	4.73	4.72	4.73	4.72	4.72	4.67	4.67	4.72	4.72
MTF-MTF-PEOMA-P	8.82	8.83	8.82	8.83	8.82	8.82	8.77	8.78	8.82	8.82
MTF-MTF-PEOMA-UP	4.72	4.73	4.72	4.73	4.72	4.72	4.67	4.67	4.72	4.72
MTF-TR-PEOMA-P	48.54	48.57	48.55	48.58	48.54	48.57	48.50	48.54	48.49	48.53
MTF-TR-PEOMA-UP	4.72	4.73	4.72	4.73	4.72	4.72	4.67	4.67	4.72	4.72
TR-MTF-PEOMA-P	8.82	8.83	8.82	8.83	8.82	8.83	8.77	8.78	8.82	8.82
TR-MTF-PEOMA-UP	4.72	4.73	4.72	4.73	4.72	4.72	4.67	4.67	4.72	4.72
TR-TR-PEOMA-P	48.54	48.57	48.55	48.58	48.54	48.57	48.49	48.53	48.50	48.54
TR-TR-PEOMA-UP	4.72	4.73	4.73	4.73	4.72	4.72	4.67	4.67	4.72	4.72

Table C.4: PEOMA-Augmented Hierarchical SLLs Results for PSE with $T = 30$ and $k = 16, 32, 64$.

D

TPEOMA-Augmented Hierarchical SLLs Results

This Appendix contains the experimental results for the OMA Hierarchical SLL on SLLs. The results contrast the performance of the MTF and TR with the hierarchical MTF-MTF-TPEOMA, TR-MTF-TPEOMA, MTF-TR-TPEOMA, and TR-TR-TPEOMA 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-TPEOMA	18.35	22.20	27.25	29.73	5.93	7.42	5.91	5.98	26.60	27.56
MTF-TR-TPEOMA	21.02	22.82	24.68	28.43	5.94	7.24	5.91	5.95	26.53	27.52
TR-MTF-TPEOMA	16.44	22.56	25.03	31.44	4.23	6.44	3.93	4.04	22.39	23.33
TR-TR-TPEOMA	18.77	22.75	26.60	31.24	4.41	6.09	3.94	4.09	22.38	23.35
$k = 4$	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-TPEOMA	29.60	29.98	28.21	28.52	5.94	6.86	12.51	12.41	16.61	17.62
MTF-TR-TPEOMA	28.16	28.27	27.98	28.36	5.94	7.17	11.51	11.78	16.73	17.76
TR-MTF-TPEOMA	32.96	32.58	31.05	31.17	4.33	5.56	11.96	12.90	14.12	15.43
TR-TR-TPEOMA	30.27	31.76	31.02	31.46	4.26	5.98	13.81	13.26	13.58	14.97
$k = 8$	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-TPEOMA	13.45	15.44	10.73	13.04	6.60	8.09	7.01	8.98	9.62	11.56
MTF-TR-TPEOMA	11.51	14.08	10.97	13.41	6.11	7.84	6.99	8.83	8.74	11.00
TR-MTF-TPEOMA	12.50	14.81	13.42	15.62	4.39	6.57	7.50	9.66	8.09	10.15
TR-TR-TPEOMA	14.80	16.88	12.38	15.07	4.37	6.54	7.64	9.87	8.30	10.67
$k = 16$	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-TPEOMA	4.05	6.49	4.27	6.59	16.17	20.28	4.21	7.46	4.53	6.65
MTF-TR-TPEOMA	4.24	6.96	4.27	6.94	4.49	7.27	4.85	8.19	4.19	6.73
TR-MTF-TPEOMA	4.94	7.24	4.52	6.80	5.73	8.03	3.91	6.95	3.81	5.97
TR-TR-TPEOMA	4.36	7.12	4.42	7.18	5.17	8.20	3.74	8.20	3.86	6.27
$k = 32$	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-TPEOMA	1.50	3.39	1.55	3.58	13.64	15.39	2.30	4.37	1.40	3.24
MTF-TR-TPEOMA	1.84	4.31	1.67	4.28	1.77	4.38	2.45	5.69	1.49	3.83
TR-MTF-TPEOMA	1.68	3.71	1.67	3.62	1.50	3.44	2.91	4.84	1.33	3.16
TR-TR-TPEOMA	1.49	3.97	1.68	4.26	1.68	4.33	1.68	5.09	1.50	3.95
$k = 64$	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-TPEOMA	0.51	1.50	0.50	1.68	12.68	12.67	0.55	1.74	0.34	1.56
MTF-TR-TPEOMA	0.50	2.47	0.50	2.49	0.50	2.90	0.55	2.76	0.39	2.18
TR-MTF-TPEOMA	0.50	1.81	0.50	1.68	0.50	1.83	0.55	1.71	0.19	1.09
TR-TR-TPEOMA	0.50	2.67	0.49	2.55	0.49	2.32	0.55	2.78	0.34	2.00

Table D.1: TPEOMA-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-TPEOMA	28.76	32.90	29.16	33.28	50.17	51.01	6.45	6.42	29.64	33.03
MTF-TR-TPEOMA	28.78	32.94	29.17	33.29	49.78	50.96	6.47	6.42	29.63	33.03
TR-MTF-TPEOMA	30.50	36.53	28.29	34.76	60.16	61.10	4.82	4.88	25.73	30.75
TR-TR-TPEOMA	30.35	36.42	28.26	34.63	60.12	61.06	4.82	4.89	25.78	30.84
$k = 4$	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-TPEOMA	52.32	55.24	53.26	56.14	45.45	48.71	17.73	17.90	51.76	54.73
MTF-TR-TPEOMA	51.91	54.84	52.74	55.73	45.12	48.42	17.85	17.89	51.47	54.53
TR-MTF-TPEOMA	56.16	58.38	56.82	59.16	49.33	52.16	23.57	23.61	55.58	57.82
TR-TR-TPEOMA	55.59	58.01	56.57	58.83	49.04	51.90	22.80	23.12	54.99	57.42
$k = 8$	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-TPEOMA	56.15	58.16	56.63	58.57	53.44	55.62	16.75	18.63	54.18	57.08
MTF-TR-TPEOMA	55.77	57.95	56.37	58.32	52.79	54.96	16.28	18.25	53.89	56.94
TR-MTF-TPEOMA	57.06	58.89	57.48	59.20	54.57	56.43	17.45	19.17	55.13	57.71
TR-TR-TPEOMA	56.69	58.54	57.14	59.03	53.64	55.76	16.78	19.08	54.98	57.65
$k = 16$	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-TPEOMA	56.10	58.30	56.06	58.31	54.41	57.11	31.71	35.22	52.69	56.54
MTF-TR-TPEOMA	58.50	59.99	58.55	60.05	56.90	55.77	31.52	35.36	54.59	57.97
TR-MTF-TPEOMA	56.00	58.28	56.31	58.45	54.64	57.18	32.16	35.67	52.57	56.61
TR-TR-TPEOMA	58.70	60.05	58.77	60.10	57.18	58.97	32.04	35.49	54.77	58.15
$k = 32$	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-TPEOMA	59.59	61.25	60.09	61.48	58.20	60.53	55.40	56.47	53.85	58.81
MTF-TR-TPEOMA	63.07	63.16	63.04	63.17	62.48	63.03	60.26	60.86	59.85	61.83
TR-MTF-TPEOMA	59.93	61.34	60.07	61.42	58.11	60.44	55.47	56.50	54.20	58.99
TR-TR-TPEOMA	63.11	63.20	63.02	63.17	62.67	63.08	60.35	60.88	60.12	61.87
$k = 64$	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-TPEOMA	62.23	62.91	62.30	62.95	59.76	62.22	61.60	62.44	16.49	52.20
MTF-TR-TPEOMA	63.38	63.36	63.26	63.30	63.00	63.29	63.20	63.30	21.75	53.45
TR-MTF-TPEOMA	62.29	62.91	62.31	62.92	59.84	62.26	61.61	62.37	17.98	50.91
TR-TR-TPEOMA	63.20	63.31	63.30	63.31	63.00	63.27	63.25	63.28	19.83	53.31

Table D.2: TPEOMA-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-TPEOMA	27.02	28.56	28.55	30.70	22.53	22.42	5.38	5.42	29.36	29.97
MTF-TR-TPEOMA	26.39	28.23	28.30	30.82	22.85	22.50	5.55	5.53	29.20	29.93
TR-MTF-TPEOMA	37.44	37.74	36.73	39.06	42.88	41.49	3.43	3.51	29.22	28.73
TR-TR-TPEOMA	37.67	37.81	36.05	38.62	43.16	41.62	3.46	3.52	29.24	28.77
MTF-MTF-TPEOMA-P	25.47	26.88	26.85	29.00	23.57	22.81	7.63	7.64	26.90	27.39
MTF-MTF-TPEOMA-UP	31.01	31.81	33.41	35.12	26.12	25.26	5.49	5.46	30.68	30.83
MTF-TR-TPEOMA-P	25.38	26.99	27.61	29.64	23.29	22.84	7.68	7.67	26.78	27.31
MTF-TR-TPEOMA-UP	30.35	31.63	32.97	34.98	25.99	25.27	5.46	5.45	30.75	30.84
TR-MTF-TPEOMA-P	36.76	36.89	36.44	37.62	43.33	42.18	5.64	5.69	26.79	26.14
TR-MTF-TPEOMA-UP	41.08	41.08	41.64	43.04	46.19	44.45	3.49	3.54	30.60	29.51
TR-TR-TPEOMA-P	50.55	57.93	47.64	54.97	63.02	64.43	5.75	5.72	43.52	52.53
TR-TR-TPEOMA-UP	41.33	40.24	42.14	43.40	46.20	44.23	3.41	3.48	30.62	29.57
$k = 4$	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-TPEOMA	25.66	24.76	25.25	24.30	21.18	20.49	8.76	8.81	23.00	22.00
MTF-TR-TPEOMA	25.75	24.94	25.29	24.45	21.50	20.63	10.35	10.35	23.05	22.11
TR-MTF-TPEOMA	26.68	25.50	25.73	24.74	24.09	23.29	12.65	12.78	23.85	22.23
TR-TR-TPEOMA	26.73	25.69	25.91	24.91	24.62	23.37	10.76	11.08	23.90	22.29
MTF-MTF-TPEOMA-P	25.60	24.17	24.90	23.54	22.18	20.87	13.02	12.84	23.19	21.24
MTF-MTF-TPEOMA-UP	42.10	39.16	41.55	38.01	35.17	31.71	13.76	13.34	37.49	32.10
MTF-TR-TPEOMA-P	25.92	24.51	25.11	23.75	22.37	21.15	12.55	12.65	23.36	21.29
MTF-TR-TPEOMA-UP	42.22	38.97	41.66	38.22	34.37	31.59	12.76	13.60	38.16	32.20
TR-MTF-TPEOMA-P	26.63	25.04	25.60	23.97	24.85	23.50	14.82	15.24	24.08	21.49
TR-MTF-TPEOMA-UP	43.18	39.62	42.10	38.69	37.53	34.19	15.44	15.38	38.51	32.52
TR-TR-TPEOMA-P	56.25	58.39	57.31	59.36	48.98	51.73	23.49	23.56	54.78	57.22
TR-TR-TPEOMA-UP	43.26	39.84	42.22	38.56	38.33	34.79	16.54	16.28	38.92	32.70
$k = 8$	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-TPEOMA	24.16	23.11	24.08	23.01	21.07	20.42	7.24	7.84	25.16	23.94
MTF-TR-TPEOMA	26.94	25.83	26.91	25.76	23.62	23.10	7.15	7.91	27.86	26.91
TR-MTF-TPEOMA	25.16	23.93	25.13	23.88	22.07	21.44	7.44	8.30	25.92	24.88
TR-TR-TPEOMA	27.99	26.71	28.20	26.86	24.54	24.04	7.32	8.42	28.95	27.84
MTF-MTF-TPEOMA-P	25.41	24.06	25.45	24.02	22.93	21.98	11.08	11.65	26.41	25.14
MTF-MTF-TPEOMA-UP	53.30	50.81	53.18	50.82	48.24	46.62	14.10	15.91	54.24	51.80
MTF-TR-TPEOMA-P	27.83	26.35	27.58	26.23	24.77	24.04	13.29	14.17	28.36	27.17
MTF-TR-TPEOMA-UP	55.93	53.34	56.00	53.17	51.45	49.43	14.54	16.85	56.75	53.76
TR-MTF-TPEOMA-P	26.60	25.14	26.72	25.16	23.90	22.89	11.09	11.82	27.53	26.08
TR-MTF-TPEOMA-UP	54.03	51.54	54.38	51.86	49.27	47.49	13.82	15.99	55.41	52.76
TR-TR-TPEOMA-P	52.31	56.01	53.08	56.51	47.45	51.55	19.82	21.41	46.80	53.71
TR-TR-TPEOMA-UP	57.12	54.28	57.35	54.30	52.44	50.07	14.74	17.01	57.83	54.82

Table D.3: TPEOMA-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-TPEOMA	23.80	23.52	24.06	23.73	22.67	22.39	13.49	14.10	24.21	23.79
MTF-TR-TPEOMA	33.02	32.61	33.16	32.83	31.81	31.74	21.38	22.18	33.01	32.58
TR-MTF-TPEOMA	23.93	23.72	24.26	23.94	22.84	22.66	13.58	14.24	24.37	24.11
TR-TR-TPEOMA	33.07	32.82	33.23	32.84	32.14	31.99	21.50	22.36	33.14	32.72
MTF-MTF-TPEOMA-P	26.09	25.58	26.33	25.87	24.97	24.51	17.12	17.48	26.45	25.90
MTF-MTF-TPEOMA-UP	58.76	58.65	58.88	58.81	57.45	57.45	43.84	44.43	58.88	58.49
MTF-TR-TPEOMA-P	37.26	36.79	37.43	36.91	36.33	35.91	32.26	32.85	37.69	37.10
MTF-TR-TPEOMA-UP	60.82	60.14	60.88	60.20	60.12	59.49	48.31	49.30	60.75	59.99
TR-MTF-TPEOMA-P	26.19	25.85	26.37	25.94	25.01	24.74	17.22	17.65	26.68	26.12
TR-MTF-TPEOMA-UP	58.89	58.83	58.98	58.97	57.83	57.60	44.11	44.74	58.97	58.61
TR-TR-TPEOMA-P	53.34	57.36	53.39	57.48	47.36	53.61	34.59	37.52	34.08	50.85
TR-TR-TPEOMA-UP	60.96	60.32	61.06	60.44	60.26	59.73	48.55	49.73	60.76	60.13
$k = 32$	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-TPEOMA	14.84	14.73	14.85	14.78	14.67	14.58	11.65	11.81	14.74	14.68
MTF-TR-TPEOMA	32.31	31.97	32.25	32.08	32.31	31.99	28.30	28.41	32.03	31.67
TR-MTF-TPEOMA	14.75	14.79	14.76	14.78	14.61	14.61	11.75	11.84	14.79	14.71
TR-TR-TPEOMA	32.13	32.02	32.15	31.97	31.91	31.87	28.30	28.39	31.67	31.62
MTF-MTF-TPEOMA-P	18.19	18.05	18.03	18.02	17.83	17.88	15.00	15.11	18.14	17.94
MTF-MTF-TPEOMA-UP	63.87	64.72	64.32	64.92	62.98	63.97	56.55	56.59	63.71	64.31
MTF-TR-TPEOMA-P	49.54	49.61	49.66	49.54	49.22	49.30	48.96	48.58	50.04	49.72
MTF-TR-TPEOMA-UP	63.06	62.78	63.15	62.80	62.95	62.62	61.63	61.46	62.94	62.63
TR-MTF-TPEOMA-P	18.11	18.06	18.06	18.02	18.10	17.84	15.04	15.06	18.15	17.95
TR-MTF-TPEOMA-UP	63.96	64.83	63.81	64.85	63.21	64.05	56.57	56.57	63.45	64.27
TR-TR-TPEOMA-P	53.29	60.84	54.12	61.06	49.76	58.62	59.43	60.37	7.25	45.85
TR-TR-TPEOMA-UP	63.05	62.79	63.12	62.82	62.96	62.65	61.68	61.42	62.97	62.69
$k = 64$	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-TPEOMA	11.98	11.77	12.08	11.80	12.03	11.76	11.64	11.29	11.87	11.67
MTF-TR-TPEOMA	57.81	56.93	57.84	56.69	58.22	56.91	57.51	56.10	57.59	56.42
TR-MTF-TPEOMA	12.04	11.78	12.00	11.74	12.03	11.77	11.56	11.25	11.95	11.70
TR-TR-TPEOMA	57.77	56.69	57.94	56.77	58.14	56.85	57.27	56.13	57.43	56.45
MTF-MTF-TPEOMA-P	16.62	16.50	16.61	16.46	16.70	16.51	17.03	16.88	16.51	16.43
MTF-MTF-TPEOMA-UP	78.73	78.46	78.99	78.62	78.31	78.04	71.94	71.98	78.08	77.72
MTF-TR-TPEOMA-P	59.08	58.56	58.45	58.37	59.11	58.34	58.38	57.83	58.28	58.10
MTF-TR-TPEOMA-UP	63.57	63.03	63.32	63.06	63.33	62.91	63.48	62.86	63.04	62.69
TR-MTF-TPEOMA-P	16.63	16.50	16.63	16.51	16.68	16.55	17.09	16.85	16.51	16.43
TR-MTF-TPEOMA-UP	78.79	78.42	78.88	78.50	78.39	77.99	71.86	71.96	62.93	62.69
TR-TR-TPEOMA-P	55.68	61.74	56.44	61.89	36.59	57.56	63.20	63.30	0.0	28.10
TR-TR-TPEOMA-UP	63.46	63.00	63.54	63.08	63.57	63.00	63.32	62.77	63.24	62.70

Table D.4: TPEOMA-Augmented Hierarchical SLLs Results for PSE with $T = 30$ and $k = 16, 32, 64$.