Optimizing Self-organizing Lists-on-Lists Using Transitivity and Pursuit-Enhanced Object Partitioning

O. Ekaba Bisong, B. John Oommen

Optimizing Self-Organizing Lists-on-Lists using Transitivity and Pursuit-Enhanced Object Partitioning

O. Ekaba Bisong[1] and B. John Oommen[2⋆]

School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6.
[1]ekaba.bisong@carleton.ca,[2]oommen@scs.carleton.ca

Abstract. The study of Self-organizing lists deals with the problem of lowering the average-case asymptotic cost of a list data structure receiving query accesses in Non-stationary Environments (NSEs) with the so-called *"locality of reference"* property. The de facto schemes for *Adaptive* lists in such Environments are the Move To Front (MTF) and Transposition (TR) rules. However, significant drawbacks exist in the asymptotic accuracy and speed of list re-organization for the MTF and TR rules. This paper improves on these schemes using the design of an Adaptive list data structure as a hierarchical data *"sub"*-structure. In this framework, we employ a hierarchical Singly-Linked-Lists on Singly-Linked-Lists (SLLs-on-SLLs) design, which divides the list data structure into an outer and inner list context. The inner-list context is itself a SLLs containing sub-elements of the list, while the outer-list context contains these sublist partitions as its primitive elements. The elements belonging to a particular sublist partition are determined using reinforcement learning schemes from the theory of Learning Automata. In this paper, we show that the Transitivity Pursuit-Enhanced Object Migration Automata (TPEOMA) can be used in conjunction with the hierarchical SLLs-on-SLLs as the dependence capturing mechanism to learn the probabilistic distribution of the elements in the Environment. The idea of *Transitivity* builds on the Pursuit concept that injects a noise filter into the EOMA to filter divergent queries from the Environment, thereby increasing the likelihood of training the Automaton to approximate the "true" distribution of the Environment. By taking advantage of the Transitivity phenomenon based on the statistical distribution of the queried elements, we can infer "dependent" query pairs from non-accessed elements in the transitivity relation. The TPEOMA-enhanced hierarchical SLLs-on-SLLs schemes results in superior performances to the MTF and TR schemes as well as to the EOMA-enhanced hierarchical SLLs-on-SLLs schemes in NSEs. However, the results are observed to have superior performances to the PEOMA-enhanced hierarchical schemes in Environments with a Periodic non-stationary distribution but were inferior in Markovian Switching Environments.

⋆ *Chancellor's Professor*; *Life Fellow: IEEE* and *Fellow: IAPR*. This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway.

Keywords: Learning Automata (LA) · Transitivity and Pursuit-Enhanced Object Migration Automaton (TPEOMA) · "Adaptive" Data Structure (ADS) · Singly-Linked-Lists on Singly-Linked-Lists (SLLs-on-SLLs).

1 Introduction

Research in self-organizing lists is aimed at mitigating the worst-case linear cost of list retrieval by adaptively rearranging the list nodes in response to query accesses from a Non-Stationary Environment (NSE). This paper optimizes the singly linked-list data structure. The models of NSEs are covered in Section 3.1. The goal of the rearrangement is to move towards the head of the list, elements that are more frequently accessed. This has the result of improving the asymptotic average cost of retrievals. However, this action requires information of the *"unknown"* probability distribution of the Environment. The Environment has a dependency property where element O_i is not conditionally independent of O_j, $O_i \not\perp O_j$. This property is called "locality of reference". Queries are requests coming from the Environment to retrieve elements from the data structure.

The asymptotic cost is an empirical measure for assessing the algorithmic performance of the list re-organization strategies [1]. This is performed by implementing the corresponding strategy, and taking an ensemble average of the averages of the cost as the algorithm converges. Since the schemes are ergodic (meaning that they have a final solution that is independent of the starting states of a Markov chain), they can be seen to provide an accurate estimation of the asymptotic cost.

The formal expression for the asymptotic cost of an adaptive strategy, A, is given as:

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

$$= \sum_{1 \le j \le J!} [\, P\{\pi_j\}_A \sum_{1 \le i \le J} s_i \pi_j(i) \,]. \tag{2}$$

In Eq. (1) and Eq. (2), 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. For more mathematical details on the asymptotic cost and Markov chains, the reader is directed to [1, 5].

The de facto schemes for list self-organization in NSEs are the Move To Front (MTF) and Transposition (TR) rules. In the MTF update scheme, the queried element is moved to the list head, except when it is the first element, because, in that case, it is already at the head (Figure 1). As opposed to this, in the TR adaptive scheme, if the queried element is not already at the list head, it is moved one position towards the front of the list (Figure 2).

Another deterministic scheme for self-organization in NSEs is the Frequency Count (FC) scheme. The FC rule maintains an accumulator for recording the access frequencies of the list elements. The resulting list is re-arranged according to

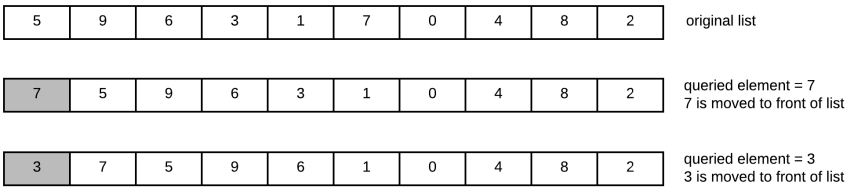


Fig. 1. A diagrammatic description of the Move-To-Front (MTF) rule

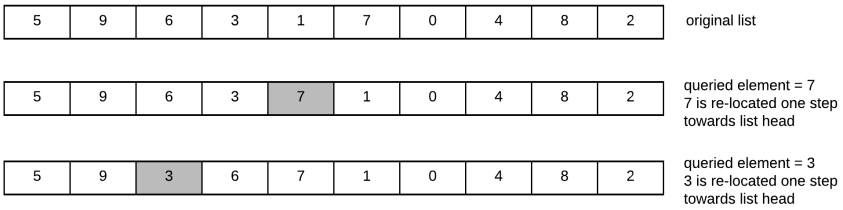


Fig. 2. A diagrammatic description of the Transposition (TR) rule

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 [4]. Notwithstanding, the FC scheme has some obvious drawbacks, the first being that the memory cost scales poorly for large lists [12]. Secondly, in environments exhibiting *"locality of reference"*, the FC scheme yields an unacceptable performance [11, 15].

The MTF and TR rules are of greater interest to this work for two reasons. The first being that they have been shown to empirically out-perform the other schemes [2], and the second, that the time and space complexities involved in implementing other surveyed schemes render most of them impractical for real-world settings. It is for these sort of Environments with dependent query accesses that this present work seeks to provide novel solutions.

Indeed, while these schemes (MTF, TR, FC) show superior performances in minimizing the asymptotic average-cost of Singly-Linked-Lists (SLLs) in NSEs [13], they, however, suffer peculiar drawbacks in NSEs exhibiting *"locality of reference"*. An empirical analysis of TR responding to query accesses from different probability distributions shows that TR outperforms MTF for the Zipf's distribution [17], and the results of [9] showed that the asymptotic cost of TR outperforms MTF for the Lotka, exponential, linear, and 80-20 probability distributions. However, the MTF has a faster adaptive rate and quickly converges early-on in the algorithm's execution [8].

This work combines the MTF and TR rules to take advantage of the quick updates of the MTF rule and the asymptotically stable convergence of the TR rule in designing our improved hierarchical adaptive strategies. This design led to the hierarchical Singly-Linked-Lists on Singly-Linked-Lists (SLLs-on-SLLs) variations of the MTF-MTF, MTF-TR, TR-MTF and TR-TR schemes, where the first component is the rule governing the list outer-context and the second component is the rule for the list inner-context [1].

However, the hierarchical SLLs-on-SLLs configuration as-is results in a certain static ordering of the elements within the inner sublist context. This presupposed static ordering carries the "false" assumption that there exists a probabilistic dependence between the elements in the sublist. In reality, the initial elements within a sublist are due to an arbitrary permutation. Hence, it turns out that the performance of the vanilla hierarchical SLLs-on-SLLs is worse of than the MTF and TR in NSEs [1, 6].

This *static* pre-supposed ordering is relaxed by the introduction of a reinforcement learning update scheme from the theory of Learning Automata (LA) called the Object Migration Automaton (OMA). The OMA algorithm is designed to learn the probabilistic dependence of elements in the Environment. This information is used to update the elements contained in the hierarchical sublist context represents their dependence ordering in the Environment. This formulation led to the OMA-hierarchical SLLs-on-SLLs consisting of the MTF-MTF-OMA, MTF-TR-OMA, TR-MTF-OMA and TR-TR-OMA schemes [1].

However, the OMA suffers from the "deadlock" problem[1] which occurs when an accessed element is swapped from one action to another and then back to the original action and thus prevented from converging to their optimal ordering (this concept is further explained in Section 2.2). To mitigate this, the Enhanced Object Migration Automaton (EOMA) was introduced by [10] that imposes conditions to restrict unnecessary swaps on action-state boundaries as well as to redefine the convergence criteria to when the automaton is in the two-innermost states as the "final" states instead of when the automaton is in the innermost state.

Bisong and Oommen [6] employed the EOMA reinforcement scheme in designing the EOMA-augmented hierarchical SLLs-on-SLLs which resulted in superior performances to the de facto MTF and TR schemes and the OMA-augmen-ted hierarchical schemes in NSEs. Further, the work by [7] further improved the performance of the hierarchical SLLs-on-SLLs by incorporating the PEOMA reinforcement scheme. The PEOMA algorithm by [18] employed the Pursuit concept to filter divergent query pairs from the Environment. To the best of our knowledge, the work by [7] is currently the state-of-the-art for self-organizing lists in NSEs.

This paper further explores the state of the art by incorporating the concept of Transitivity in the PEOMA algorithm, that is the Transitivity Pursuit-Enhanced Object Migration Automaton (TPEOMA) to design a TPEOMA-

[1] Albeit referred to as a "deadlock" in the literature, it could more appropriately be described as a "livelock".

augmented hierarchical SLLs-on-SLLs. This proposed hierarchical formulation consists of the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA schemes. The Transitivity concept is used to infer "dependent" queries from the Environment to further train the automaton to approximate its "unknown" distribution. The Transitivity concept is discussed in more detail in Section 2.3.

The novel contributions of this paper include:

- The design and implementation of the TPEOMA-enhanced SLLs-on-SLLs;
- Demonstrating the superiority of the TPEOMA-augmented hierarchical schemes to the MTF and TR rules and to the original OMA-augmented schemes that pioneered the idea of a hierarchical LOL approach;
- Demonstrating the superiority of the TPEOMA-augmented hierarchical schemes to the EOMA-augmented hierarchical schemes;
- Highlighting the performances of the TPEOMA-augmented hierarchical schemes to the PEOMA variants;
- Showing that the "Periodic" and "UnPeriodic" versions of the TPEOMA-augmented hierarchical schemes yielded superior and comparable performances respectively in PSEs to those without such additions.

Section 1 introduces the idea of a hierarchical SLLs-on-SLLs for minimizing the asymptotic average case for list retrieval in NSEs. It also lays out the case for incorporating the OMA-family of reinforcement schemes for learning the probability dependence distribution of elements in the Environment. Section 2 reviews the theory of LA[2] as the foundational theory for the TPEOMA reinforcement scheme. In addition, this section accesses the idea of the Transitivity concept that builds on the Pursuit concept in the PEOMA algorithm. Section 3 explains the design of the TPEOMA-augmented hierarchical SLLs-on-SLLs for NSEs. Section 4 presents the Results and Discussions, and Section 5 concludes the paper.

2 Theoretical Background

2.1 Learning Automata

Learning automata (LA) arose in the Soviet Union in the 1960s by Tsetlin as a computational adaptive scheme for learning. [20] The LA task can be modeled by means of a feedback loop between the Environment and the automaton. The automaton interacts with the Environment by choosing from a set of actions based on the feedback it receives from the Environment. The LA model is set up as an adaptive process [3] 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 [1, 5, 18].

[2] Due to space limitations, the background material is only briefly surveyed. The seminal work by [14] and the theses by Shirvani [18] and the first author of this paper [5] contain exhaustive details of the theory and applications of LA.

2.2 The OMA & EOMA

The OMA improves on the Tsetlin/ Krinsky strategies for partitioning objects into groups in the Equal-Partitioning Problem (EPP) [16]. In the OMA, the number of actions represents the number of groups, or partitions, R, where each action contains a set number of states, N. The OMA partitions the object group W into R partitions by moving the abstract objects $\mathcal{O}$ around the action-states of the automaton. To learn more about the OMA algorithm the reader is referred to [5–7, 18].

The EOMA algorithm improves on the OMA to mitigate the inability of the OMA algorithm to converge due to a "deadlock" scenario. The deadlock scenario occurs when there is a query pair $\langle O_i, O_j \rangle$ in a stream of query pairs belonging to different actions, α_h and α_g. 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. The boundary state of the OMA is the outermost memory state of an action α (see Figure 3). To learn more about the EOMA algorithm the reader is referred to [5–7, 18].

To resolve the deadlock scenario for the query pair $\langle O_i, O_j \rangle$, let us say that there exists an object in the boundary state of the action-group, α_g containing O_j, given that the other element O_i is not in the boundary state of its action-group α_h. The EOMA moves moves O_j from action-group α_g to be in the boundary state of action-group, α_h. By taking this step, the partitions become unequal. To regain the equi-partitioning, the object that is closest to O_i in α_h, which we will call O_l, is moved to the boundary state of α_g.

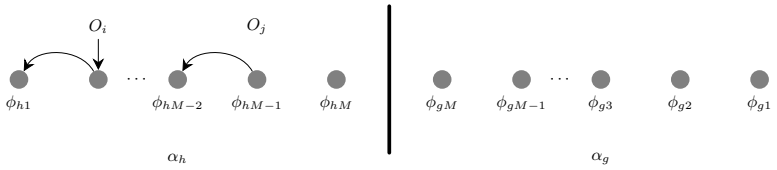
Hence, given a set of query elements, the transitions of the EOMA for the abstract objects $\langle O_i, O_j \rangle$ on reward and on penalty are illustrated in Figure 3. In addition, the EOMA also modifies the convergence criteria to reduce its vulnerability to divergent queries by setting the two-innermost states as the "final" states, as opposed to just the innermost state in the vanilla OMA.

To learn more about how the EOMA algorithm mitigates the "deadlock" scenario in the OMA, the reader is referred to [5–7, 18].

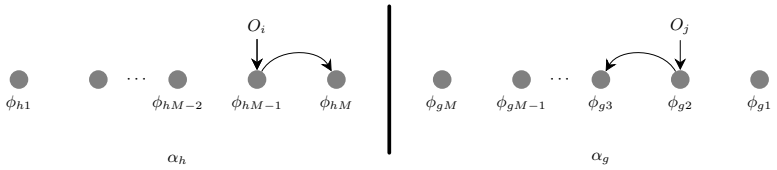
2.3 The Case for Transitivity

The Pursuit concept is incorporated into the EOMA design to filter *divergent* queries from the Environment using Maximum Likelihood Estimates (MLEs). It works by updating the joint query probabilities using ranked estimates of the reward probabilities to asymptotically choose or "pursue" better actions. For a detailed discussion of the pursuit concept, the reader is referred to [18]. The Pursuit-EOMA scheme was the best-known object-partitioning algorithm until the authors of [19] introduced the TPEOMA.

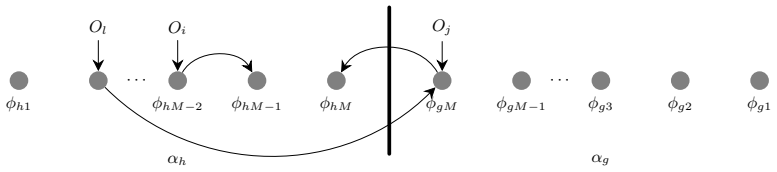
The TPEOMA algorithm is based on the observation that the Pursuit matrix can also be used to infer underlying relations in the Environment [18]. It then invokes a policy that spins off reward/penalty operations by incorporating the statistics obtained between objects that have been previously accessed. The Pursuit matrix M is defined as a $\frac{W}{R} \times \frac{W}{R}$ matrix whose element $[i, j]$ are



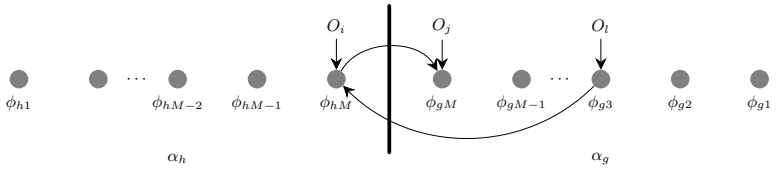
(a) On reward: Move the accessed abstract objects $\langle O_i, O_j \rangle$ towards the extreme states.



(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.

Fig. 3. The EOMA Algorithm

the probabilities that $\langle O_i, O_j \rangle$ are simultaneously accessed where $1 \leq i, j \leq W$. The Pursuit matrix M is computed using Maximum Likelihood estimates, where each entry m_i in the matrix is the ratio of the frequency of occurrence of m_i to the total number of query accesses. The probabilities in M sum to unity. The reader is referred to [19] for a details analysis of the Pursuit matrix.

When an estimate of the Pursuit matrix is obtained, the transitivity property can be used to *infer* queries to further train the automaton to learn the model of dependence of the Environment. In other words, if the current query received from the Environment is $\langle O_i, O_j \rangle$ and O_j is in relation with $O_k, k \in \{1, \cdots, W_j\}$, $k \neq i$, we can infer that O_i is also in a relation with O_k, where i, j, and k are

the indices of the entries in the Pursuit matrix. Thus, given the transitivity threshold, τ_T, which is a suitable user-defined threshold which is set to a value $\frac{1}{W^2-W}$, where W is the number of elements in the list. We 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 transitivity relation is illustrated in Figure 4. The reader is referred to [19] for a thorough analysis of the Transitivity property in the TPEOMA algorithm.

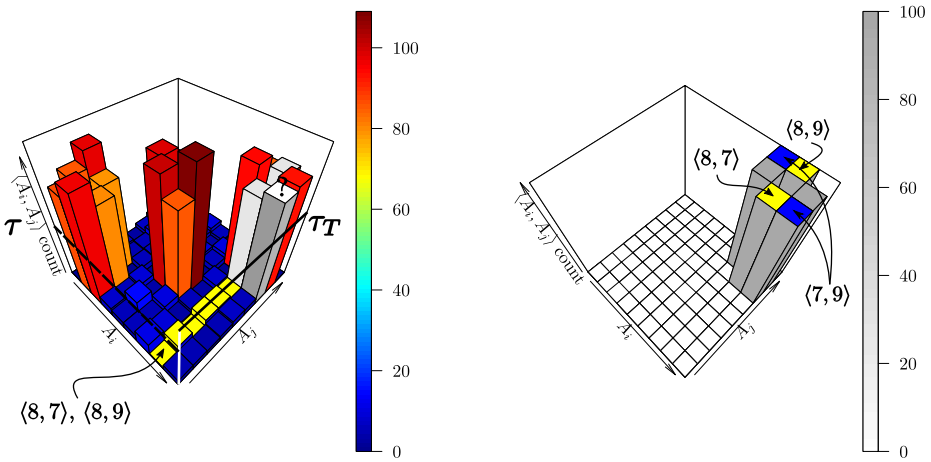


Fig. 4. Left: This figure displays the magnitude of the elements in the Transitivity-Pursuit matrix showing the joint probability distribution of query pairs with a defined cut-off threshold τ and transitivity threshold τ_T. Right: The transitivity relation is demonstrated in the simplest case. Source: [18]

3 TPEOMA-Augmented Hierarchical SLLs-on-SLLs

The concept of a hierarchical data "sub"-structure involves dividing a list of size W 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.

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.

In NSEs with *"locality of reference"*, let us assume that query accesses m are made to a sub-context (or local context) Q_a. For a given query access 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.

In the TPEOMA-Augmented Hierarchical SLLs-on-SLLs, 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. The augmentation of hierarchical SLLs with the TPEOMA reinforcement scheme gives rise to the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR- MTF-TPEOMA and the TR-TR-TPEOMA.

3.1 Models of NSEs

In NSEs, the penalty probabilities for each action vary with time. In the context of 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. The *"locality of reference"* occurs when there exists a probabilistic dependence between the consecutive queries [2]. 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 NSEs and their 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 W, divided into k sub-lists, with each sub-list containing $\frac{W}{k}$ 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.

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 sub-list 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.

4 Results and Discussions

The experimental setup for the simulations involving the TPEOMA-augmented hierarchical schemes in MSEs involved splitting a list of size 128 into k sublists with $k \in 2, 4, 8, 16, 32, 64$. The degree of dependence of the MSE, α, was set to 0.9 and the period for the PSE, $T = 30$. 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 aforementioned query generators. For conciseness sake, we present results for $k = 8$.

From Table 1, when $k = 8$, we observed that the TPEOMA-augmented hierarchical schemes were superior to the MTF and TR standalone schemes for all query distributions in the MSE. 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 (6.60, 6.11, 4.39, 4.37) and (8.09, 7.84, 6.57, 6.54) respectively. As opposed to this, the corresponding asymptotic and amortized costs for the MTF and TR were significantly higher at (39.30, 48.25) and (39.17, 48.66) respectively. Further, in the Exponential distribution for the MSE, the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA had asymptotic and amortized costs of (7.01, 6.99, 7.50, 7.64) and (8.98, 8.83, 9.66, 9.87), while the MTF and TR rule had asymptotic and amortized costs of (8.72, 10.52) and (8.71, 10.93) respectively. Hence, showing the superiority of the the TPEOMA-augmented schemes to the MTF and TR rules respectively in the MSE. Also, from Table 1, observe that while the TPEOMA-augmented hierarchical schemes are superior to the EOMA-augmented hierarchical schemes, the PEOMA-augmented hierarchical still boasts superior performances in MSEs.

Figure 5 accesses the asymptotic cost ratio of the MTF-MTF-TPEOMA scheme to the MTF for varying number of sublist partitions ranging from $k = \{2, 4, 8, 10, 16, 32, 64\}$. For all the query distributions under consideration, the MTF-MTF-TPEOMA scheme possesses a superior asymptotic cost to the MTF except for the Exponential scheme where the MTF has a better asymptotic cost when $k = \{2, 4\}$.

Table 1: Asymptotic (top) and Amortized (bottom) costs in **MSE** with $\alpha = 0.9$ and $k = 8$.

Scheme	Zipf	80-20	Lotka	Exp.	Linear
MTF	43.35	43.76	39.30	8.72	43.60
TR	55.44	56.74	48.25	10.52	56.79
MTF-MTF-EOMA	19.14	19.23	18.70	12.34	19.31
MTF-TR-EOMA	27.80	27.77	27.17	16.89	28.04
TR-MTF-EOMA	18.84	18.99	18.37	12.87	18.96
TR-TR-EOMA	27.55	27.62	26.96	17.17	27.70
MTF-MTF-PEOMA	5.80	6.73	1.25	2.45	6.76
MTF-TR-PEOMA	5.35	6.21	1.25	2.97	6.77
TR-MTF-PEOMA	4.67	6.00	0.96	2.88	5.61
TR-TR-PEOMA	5.07	6.49	0.98	2.99	6.34
MTF-MTF-TPEOMA	13.45	10.73	6.60	7.01	9.62
MTF-TR-TPEOMA	11.51	10.97	6.11	6.99	8.74
TR-MTF-TPEOMA	12.50	13.42	4.39	7.50	8.09
TR-TR-TPEOMA	14.80	12.38	4.37	7.64	8.30
MTF	43.25	43.82	39.17	8.71	43.64
TR	55.85	56.96	48.66	10.93	57.26
MTF-MTF-EOMA	19.35	19.40	19.26	12.90	19.45
MTF-TR-EOMA	27.93	28.02	27.54	16.57	28.08
TR-MTF-EOMA	19.09	19.18	19.07	13.35	19.18
TR-TR-EOMA	27.72	27.80	27.25	17.10	27.87
MTF-MTF-PEOMA	6.97	7.77	2.32	4.00	7.79
MTF-TR-PEOMA	7.14	7.87	2.63	4.23	8.31
TR-MTF-PEOMA	5.95	7.13	2.04	4.65	6.71
TR-TR-PEOMA	6.84	8.05	2.29	4.77	7.91
MTF-MTF-TPEOMA	15.44	13.04	8.09	8.98	11.56
MTF-TR-TPEOMA	14.08	13.41	7.84	8.83	11.00
TR-MTF-TPEOMA	14.81	15.62	6.57	9.66	10.15
TR-TR-TPEOMA	16.88	15.07	6.54	9.87	10.67

Table 2: Asymptotic (top) and Amortized (bottom) costs in **PSE** with $T = 30$ and $k = 8$.

Scheme	Zipf	80-20	Lotka	Exp.	Linear
MTF	49.64	50.24	44.52	8.46	50.08
TR	55.65	56.91	48.51	11.18	57.19
MTF-MTF-EOMA	14.63	14.70	14.12	8.59	14.72
MTF-TR-EOMA	25.82	25.90	25.32	13.88	25.92
TR-MTF-EOMA	14.39	14.49	13.76	8.92	14.50
TR-TR-EOMA	25.58	25.69	24.97	13.70	25.70
MTF-MTF-EOMA-P	7.16	7.24	6.66	6.14	7.26
MTF-MTF-EOMA-UP	7.69	7.78	7.19	8.90	7.79
MTF-MTF-PEOMA	11.80	11.29	10.57	4.10	10.30
MTF-TR-PEOMA	23.31	23.22	21.64	5.56	21.42
TR-MTF-PEOMA	12.00	11.04	10.21	4.32	10.04
TR-TR-PEOMA	20.94	20.11	19.28	5.56	21.17
MTF-MTF-PEOMA-P	7.13	7.21	6.53	6.10	7.21
MTF-MTF-PEOMA-UP	7.40	7.57	5.45	6.31	7.49
MTF-MTF-TPEOMA	12.30	12.16	12.40	11.57	11.50
MTF-TR-TPEOMA	12.79	12.62	12.61	10.37	12.03
TR-MTF-TPEOMA	12.40	12.26	12.48	10.60	12.26
TR-TR-TPEOMA	12.56	12.41	12.37	11.00	12.18
MTF-MTF-TPEOMA-P	8.95	9.19	9.56	12.56	10.49
MTF-MTF-TPEOMA-UP	12.25	11.91	13.65	12.50	14.50
MTF	49.62	50.23	44.53	8.48	50.06
TR	56.09	57.28	48.91	11.58	57.60
MTF-MTF-EOMA	14.76	14.84	14.31	8.68	14.84
MTF-TR-EOMA	25.93	26.01	25.44	12.49	26.02
TR-MTF-EOMA	14.54	14.62	14.03	9.69	14.63
TR-TR-EOMA	25.71	25.80	25.12	13.11	25.80
MTF-MTF-EOMA-P	7.28	7.38	6.82	7.53	7.40
MTF-MTF-EOMA-UP	7.86	7.95	7.48	10.57	7.92
MTF-MTF-PEOMA	12.44	11.35	10.81	4.88	10.37
MTF-TR-PEOMA	23.84	23.78	21.64	5.90	21.63
TR-MTF-PEOMA	12.59	12.13	10.41	5.25	10.13
TR-TR-PEOMA	17.33	17.05	19.32	6.32	21.22
MTF-MTF-PEOMA-P	7.27	7.34	6.75	7.07	7.36
MTF-MTF-PEOMA-UP	7.44	7.68	5.51	6.90	7.59
MTF-MTF-TPEOMA	12.63	12.50	12.49	12.34	11.92
MTF-TR-TPEOMA	12.72	12.57	12.51	12.13	11.96
TR-MTF-TPEOMA	12.69	12.54	12.51	12.00	11.97
TR-TR-TPEOMA	12.65	12.50	12.50	11.93	11.98
MTF-MTF-TPEOMA-P	9.55	9.47	9.50	10.12	10.35
MTF-MTF-TPEOMA-UP	13.33	13.08	13.09	14.86	15.22

In Table 2, the performance of the TPEOMA-augmented hierarchical schemes in the PSEs were superior to the standalone MTF and TR rules for all distributions under consideration except for the Exponential scheme which boasted slightly comparable results. As an example, consider the 80-20 distribution where the MTF-MTF-TPEOMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA had asymptotic and amortized costs of (12.16, 12.62, 12.26, 12.41) and (12.50, 12.57, 12.54, 12.50) respectively. Whereas the MTF and TR had

asymptotic and amortized costs of (50.24, 56.91) and (50.23, 57.28) respectively showing the superiority of the TPEOMA-augmented schemes in such Environments. Moreover, the results from Table 2 also indicate that the TPEOMA-augmented schemes have superior performances to EOMA-augmented and PEOMA-augmented hierarchical schemes in PSEs for the query distributions under consideration.

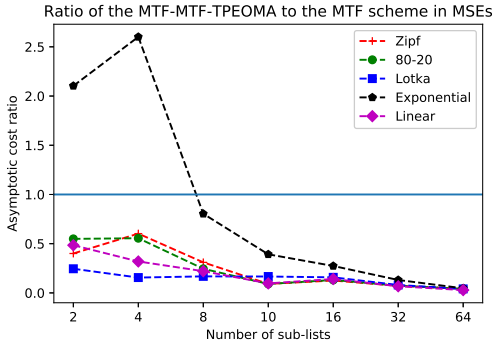
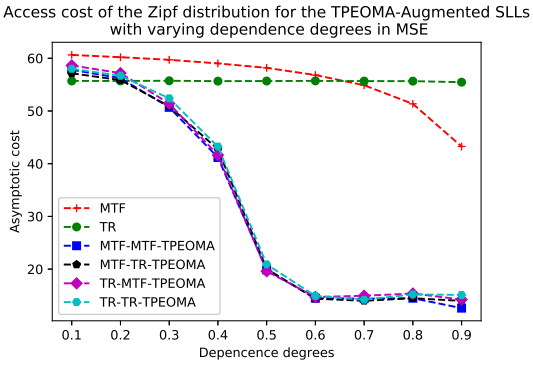


Fig. 5. The asymptotic cost ratio of the MTF-MTF-TPEOMA to the MTF scheme for different values of the sub-list partitions $k = \{2, 4, 8, 10, 16, 32, 64\}$ for MSE-dependent query Environments in which $\alpha = 0.9$.



!h

Fig. 6. 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.

Also, when the knowledge of the Periodic state change is passed to the TPEOMA-augmented hierarchical schemes, we observed that the *"periodic"*

variations yielded superior performances to their vanilla versions, whereas the *unperiodic* variations had comparable performances to the vanilla versions.

From Figure 6, we see that the TPEOMA-augmented hierarchical schemes were superior to the MTF and TR schemes in noisy Environments 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.

5 Conclusion

In this paper, we designed a TPEOMA-augmented hierarchical Singly-Linked-Lists on Singly-Linked-Lists (SLLs-on-SLLs) using reinforcement learning schemes from the theory of Learning Automata, which led to the MTF-MTF-TPE-OMA, MTF-TR-TPEOMA, TR-MTF-TPEOMA and TR-TR-TPEOMA schemes. The TPEOMA-augmented hierarchical schemes showed superior performances to the standalone MTF and TR schemes in MSEs. Also, they are superior to the EOMA-augmented hierarchical schemes. However, the PEOMA-augmented hierarchical schemes are still superior to the TPEOMA-augmented hierarchical schemes in the MSE. In the PSE, the TPEOMA-augmented hierarchical were for the most part superior to the MTF and TR schemes for the query distributions under consideration. Further, the TPEOMA-augmented schemes were also superior to the EOMA-augmented and PEOMA-augmented hierarchical schemes in the PSE. However, when the knowledge of the periodic state change were incorporated to the hierarchical schemes, the "periodic" case had superior performances to their vanilla versions while the "unperiodic" case had comparable performances.

References

1. Amer, A.: Adaptive list organizing strategies for non-stationary distributions (2004)
2. Bachrach, R., El-Yaniv, R., Reinstadtler, M.: On the competitive theory and practice of online list accessing algorithms. Algorithmica **32**(2), 201–245 (2002)
3. Bellman, R.: Adaptive control processes: A guided tour. Princeton University Press, Princeton, N.J. (1961)
4. Bentley, J.L., McGeoch, C.C.: Amortized analyses of self-organizing sequential search heuristics. Communications of the ACM **28**(4), 404–411 (1985)
5. Bisong, E.O.: On Designing Adaptive Data Structures with Adaptive Data "Sub"-Structures. Master's thesis, Carleton University (2018)
6. Bisong, E.O., Oommen, B.J.: Optimizing self-organizing lists-on-lists using enhanced object partitioning. 15th International Conference on Artificial Intelligence Applications and Innovations (2019)
7. Bisong, E.O., Oommen, B.J.: Optimizing self-organizing lists-on-lists using pursuit-oriented enhanced object partitioning. 15th International Conference on Intelligent Computing (2019)
8. Chassaing, P.: Optimality of move-to-front for self-organizing data structures with locality of references. The Annals of Applied Probability pp. 1219–1240 (1993)

9. Dong, J.: Time reversible self-organizing sequential search algorithms. (1998)
10. Gale, W., Das, S., Yu, C.T.: Improvements to an algorithm for equipartitioning. IEEE Transactions on Computers **39**(5), 706–710 (1990)
11. Hester, J.H., Hirschberg, D.S.: Self-organizing linear search. ACM Computing Surveys (CSUR) **17**(3), 295–311 (1985)
12. Knuth, D.E.: The art of computer programming, vol. 3. Pearson Education (1997)
13. McCabe, J.: On serial files with relocatable records. Operations Research **13**(4), 609–618 (1965)
14. Narendra, K.S., Thathachar, M.A.L.: Learning automata: an introduction. Courier Corporation (2012)
15. Oommen, B.J., Dong, J.: Generalized swap-with-parent schemes for self-organizing sequential linear lists. In: International Symposium on Algorithms and Computation. pp. 414–423. Springer (1997)
16. Oommen, B.J., Ma, D.C.Y.: Deterministic learning automata solutions to the equipartitioning problem. IEEE Transactions on Computers **37**(1), 2–13 (1988)
17. Rivest, R.: On self-organizing sequential search heuristics. Communications of the ACM **19**(2), 63–67 (1976)
18. Shirvani, A.: Novel Solutions and Applications of the Object Partitioning Problem. Ph.D. thesis, Carleton University, Ottawa (2018)
19. Shirvani, A., Oommen, B.J.: The advantages of invoking transitivity in enhancing pursuit-oriented object migration automata. (2017)
20. Tsetlin, M.L.: Finite automata and models of simple forms of behaviour. Russian mathematical surveys **18**, 1–27 (1963)