On Enhancing the Object Migration Automaton using the *Pursuit* Paradigm*

Abdolreza Shirvani†, and B. John Oommen‡

Abstract

One of the most difficult problems that is all-pervasive in computing is that of partitioning. It has applications in the partitioning of databases into relations, the realization of the relations themselves into sub-relations based on the partitioning of the attributes, the assignment of processes to processors, graph partitioning, and the task assignment problem etc. The problem is known to be NP-hard. The benchmark solution for this for the Equi-partitioning Problem (EPP) has involved the classic field of Learning Automata (LA), and the corresponding algorithm, the Object Migrating Automata (OMA) has been used in all of these application domains. While the OMA is a fixed structure machine, it does not incorporate the *Pursuit* concept that has, recently, significantly enhanced the field of LA. In this paper, we pioneer the incorporation of the *Pursuit* concept into the OMA. We do this by a non-intuitive paradigm, namely that of *removing* (or discarding) from the query stream, queries that could be counter-productive. This can be perceived as a filtering agent triggered by a pursuit-based module. The resulting machine, referred to as the Pursuit OMA (POMA), has been rigorously tested in all the standard benchmark environments. Indeed, in certain extreme environments it is almost *ten* times faster than the original OMA reported in the legacy papers[1] The application of the POMA to these application domains is extremely promising.

Keywords: *Object Partitioning, Learning Automata, Object Migration Automaton, Partitioning-based Learning*

1 Introduction

We are living in an era in which data is being produced at an increasing, and almost unbelievable rate. The availability of such an enormous amount of data is an invaluable source of information which can be utilized in inference, and in the analysis and understanding of the diverse domains from which the data originates, and of their associated events. The business, political and health-related advantages of possessing and utilizing these are, of course, obvious. Often the data comes from physical observations generated by sensors, which collect, for example, geological measurements or medical information. The truth of the matter, though, is that the raw data, in and of itself, has little or no value unless it is processed and transfered into information from which "knowledge"

*The second author is grateful for the partial support provided by NSERC, the Natural Sciences and Engineering Research Council of Canada. We are extremely grateful to the anonymous Referees of the earlier version of this paper for their feedback. Their input significantly improved the quality of this current version.

†This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. E-mail: AbdolrezaShirvani@cmail.carleton.ca.

‡Author's status: *Chancellor's Professor*, *Fellow: IEEE* and *Fellow: IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. The author is also an Adjunct Professor with University of Agder, Grimstad, Norway. E-mail: oommen@scs.carleton.ca.

[1]In our improved implementation of these methods (explained in the body of the paper), our current scheme is nearly *twice* as fast.

can be learned. The preprocessing of the data is more challenging than it appears because it arrives in a variety of forms and dimensions.

Considering the massive volume and unstructured nature of the data available in the majority of applications, the task of transforming it into meaningful pieces of information that can be processed, is far from trivial. Although preprocessing such large amounts of data initially seems intractable, many statistical and mathematical approaches have been developed to remedy this, and also to introduce more structure into the almost-infinite amount of unstructured data.

One effective approach by which this can be achieved is to seek for the possible underlying structures or patterns, and to try to find "regularities" embedded in such large monolithic data. This is the arena in which we operate, and the fundamental tool is that of "partitioning".

1.1 Partitioning

Partitioning real-life objects into certain subgroups, which, in turn, meet specific criteria, is a truly fundamental problem. We initiate discussion by introducing an abstract concept and then proceed to relate it to real world physical entities. As an example, in numerous applications, one could desire to group the objects into subgroups so that those possessing similar characteristics are clustered together. Usually, these characteristics are dynamic, and vary within the Environment. These similarities can also be rather nebulous. This is the problem that we investigate.

Consider the problem of partitioning a set $\mathcal{A} = \{A_1, \cdots, A_W\}$ of W physical objects into R groups $\Omega = \{G_1, \cdots, G_R\}$. We assume that the true but unknown state of nature, Ω^*, is a partitioning of the set $\mathcal{A}$ into mutually exclusive and exhaustive subsets $\{G_1^*, G_2^*, \cdots G_R^*\}$. The composition of the $\{G_i^*\}$ is unknown and the elements in the subsets fall together based on some criteria which may be mathematically formulated, or which may be subjective, or even rather, ambiguous. These objects are now presented to a learning algorithm in some manner, for example, in pairs or tuples. The goal of the algorithm is to partition $\mathcal{A}$ into a learned partition, Ω^+. The hope is to have Ω^+ converge to Ω^*.

In most cases, the underlying partitioning of Ω^* is not known, nor are the joint access probabilities by which the pairs (or the tuples of $\mathcal{A}$) are presented to the learning algorithm known. This problem is known to be NP-hard [**?**]. Clearly, if we increase the number of objects, the number of partitions increases, and in addition to this quantity, the problem's complexity grows exponentially.

With a little insight, the reader will observe that the partitioning problem is akin to clustering. This is because in clustering, one attempts, in an unsupervised manner, to gather the elements that really "should belong together". The partitioning problem has been the subject of research for more than five decades. Being *NP*-hard, a general polynomial-time solution cannot be obtained. Consequently, all of the reported solutions take advantage of various Artificial Intelligence (AI)-based heuristics.

Unarguably, AI is an extremely extensive field. The subfield of AI that we shall use to solve the partitioning problem is referred to as Learning Automata (LA). This is a relatively new field and the research in it has been limited to the span of three or four decades. We shall aim to resolve the partitioning problem using the tools available in LA.

1.2 The Object Partitioning Problem

If there exists a mutual relation between the real-world objects in the semantic domain, $\mathcal{A}$ (introduced in **??**), and a domain of abstract objects $\mathcal{O} = \{O_1, \cdots, O_W\}$, we define the partitioning of $\mathcal{O}$ in such way that the corresponding partitions of the set $\mathcal{O}$ map onto the partitions of the real, physical objects in $\mathcal{A}$ so as to mimic the state-of-nature of the real world. In other words, while we operate on and conceptually migrate the abstract objects in the set $\mathcal{O}$, the objects in $\mathcal{A}$ are not necessarily moved because they constitute real-life objects which cannot easily be moved in a real-time manner. To achieve this, it is possible to explore all partition combinations, use a ranking index, and thereafter to only seek plausible partitions, or to choose the best partition, etc. The most significant aspect of the OPP is that of identifying the *best* or *most likely realizable* partitioning. This requires the AI algorithm to perceive the semantic physical world aspects of the given objects, and to thereafter make local decisions based on the best partition in the *abstract* domain [**?**].

For example, in database systems, $\mathcal{A}$ can represent the set of records in the database or the files maintained on different servers. These are, typically, to be partitioned based on their joint access probabilities. The ideal partitioning would be to keep the most-likely records or files in the same partition (or server). The OPP can be directly applied to resolve this problem, which can be considered as a record or a file clustering problem [**?**] and [**?**]. Our aim is to devise a single comprehensive solution that will be applicable for all such real-life problems. However, rather than manipulating the real-life objects, the algorithm will run in the background, and will migrate (or move) the corresponding abstract objects.

The problem described above was merely mentioned conceptually. However, we are not aware of any application software that has *actually* incorporated a solution of the OPP into their physical realization. We believe that this is because of the complexity of the underlying NP-hard problem, and of the solutions that the various researchers proposed. This was, indeed, the case till the mid-1980's at which juncture Oommen and Ma proposed the Object Migrating Automata (OMA) to solve a special case of the OPP, namely the Equi-Partitioning Problem (EPP). The introduction of the OMA solution made real-life applications possible. The OMA resolved the EPP both efficiently and accurately, and it could thus be easily incorporated into *many* real-life application domains.

1.3 The Environment, $\mathbb{E}$, for the Queries

In order to asses the partitioning accuracy as well as the convergence speed of any EPP solution, there must be an "oracle" with a pre-defined number of classes, and with each class containing an equal number of objects. The underlying distribution of the objects among the classes is, of course, unknown to the OMA, and its goal is to migrate the objects between its classes, using the incoming queries specified by $\mathbb{E}$. This should be done in such a way that the partitioning error is minimized as the queries are encountered. Such an Environment and its associated query generating system can be characterized by three main parameters:

- The number of objects, specified by W,
- The number of groups or partitions, specified by R, and
- A quantity 'p' , which is a probability coefficient specifying the certainty by which $\mathbb{E}$ pairs the elements in the query.

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

We assume that $\mathbb{E}$ generates an "unending" continuous stream of query pairs.

By knowing how the pairs are generated, we will now proceed to visualize the partitioning problem and the solution that is offered by the OMA (or for that matter, any solution to the EPP). **??** displays three classes named G_1, G_2 and G_3, and the objects inside them are represented by integers in $\{1, \cdots, 9\}$. The original distribution of the objects between the classes is shown in **??**, at the top. This is the true unknown state of nature, i.e., Ω^*. The OMA is initialized in a purely random manner by the numbers within the range. This step is depicted at the bottom of **??**, and Ω_0 indicates the initial state of the OMA. At every iteration, a pair given by $\mathbb{E}$ is processed by the OMA, and it performs a learning step towards its convergence. The goal of the algorithm is for it to converge to a state, say Ω^+. In an optimal setting, we would hope that Ω^+ is identical to Ω^*.

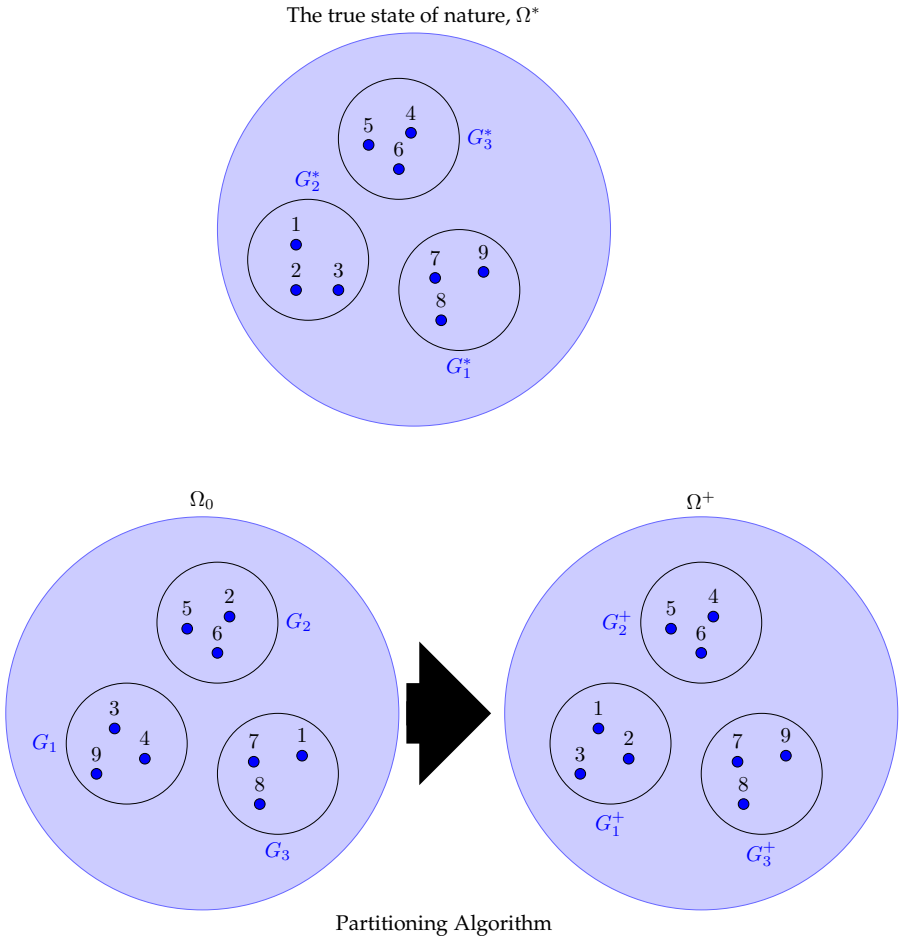


Figure 1: A figure describing the partitioning of the objects.

1.4 Paper Organization

The remainder of the paper is laid out as follows. **??** presents a survey of the field including an overview of the OPP and EPP (in **??**), and the prior "legacy" solutions that were proposed to solve them, including a Hill Climbing method and the Basic Adaptive Method. **??** describes the Object Migrating Automaton (OMA) which has been the benchmark LA-based technique that has set the standard for almost three decades. This is followed by the theoretical results about the underlying Environment whose properties we intend to pursue. In **??** we explain the rational for how the OMA *can* be improved by introducing the *Pursuit* phenomenon that has recently enhanced LA families, and it also includes the incorporation of the Pursuit concept into the OMA, that leads us to the Pursuit OMA (POMA). **??** describes the experimental results that we have obtained for the OMA and the POMA. Section **??** includes some discussions about these results. Section **??** concludes the paper.

2 Survey of the Field

2.1 Prior Solutions to the OPP/EPP

Several solutions[2] have been proposed by researchers to solve the OPP. The most elementary solution to the OPP is to estimate the query statistics [**?**], e.g. to partition the objects into different categories inferred from the frequency counts obtained from a sequence of queries for every category. The computational cost of this method grows exponentially as we increase the number of partitions. To address this issue, various heuristic solutions have been proposed in the literature, e.g., [**?**] and [**?**]. Since the OPP is a known NP-hard problem [**?**], the optimal partitioning algorithm will have an exponential computational cost [**?**], and it will be tantamount to an exhaustive search scheme. Consequently, heuristics are widely used to accelerate the search, and to thus limit the exponential growth of the problem size. Although the interpretation of the obtained results will be, unfortunately, limited and sub-optimal, the results reported in the literature are remarkably accurate and fast.

The "Hill-climbing" technique proposed by the authors of [**?**] and [**?**] is an example of one such approach. Although impractical in real-life, the advantage of these approximate solutions are their use in the validation of the correctness of the respective heuristics.

Indeed, it is well known that the hill-climbing paradigm has been widely used in the literature for local optimization. In [**?**] and [**?**] Hammer *et al* have used these principles, associated with a step-wise approach, to address the problem of attribute partitioning [**?**], and to obtain a sub-optimal partition. The heuristic is composed of two components: The Partition Evaluator (PE) and the Heuristic Searcher (HS). The PE calculates the physical "characteristic fitness" of any partition of a set of attributes and appraises how it is used. It evaluates the processing cost of any given transaction that uses this partition, and this is referred to as the Transaction Cost Estimator (TCE). The PE and HS define the two fundamental modules for examining the search space, using which it finds a near-optimal solution based on the available users' transactions. The details of these methods and the experimental results are omitted and can be found in [**?**]. The authors of [**?**] claimed that running a real-life simulation was not practical. They referred the reader to the papers [**?**] and [**?**] where some experimental results obtained by using the heuristics were provided. We too refer the reader to these references.

[2]The survey of the field of LA, the OPP and the OMA are necessarily brief. A more detailed exegesis is found in the doctoral thesis of the first author [**?**].

As opposed to the above, the BAM is based on the concept that clustering a group of objects into relevant classes can be carried out by observing a sequence of queries provided by the Environment. The BAM is an appealing method due to its simple yet intriguing properties. It also possesses advantages over its predecessor solutions to the OPP. According to the authors of [**?**] and [**?**], these advantages are:

a) The BAM is a unifying approach that can be applied to many other problems such as attribute partitioning, record clustering etc., with minor modifications;

b) The BAM does not involve collecting any query statistics;

c) It is an adaptive method which is based on simple clustering-like phenomena that operate on a line;

d) The scheme has been mathematically analyzed for simple distributional patterns.

Prior to the LA-based solutions discussed later, the BAM was the best-reported algorithm for the OPP. We refer the reader to [**?**] and [**?**] for further details of this scheme, and for a more thorough analysis.

2.2 Survey of Learning Automa

From an intuitive perspective of behavioral psychology, the "subject" is often presented with a "stimulus", and the response to the stimulus produces the "behavior". Thus, without a stimulus one cannot assume the existance of a behavior. An Automaton, by definition, models an autonomous agent, whose *behavior* manifests as a consequence of the interplay between a sequence of stimuli from the Environment. The Automaton responds adaptively to the Environment and enforces the actions which fit the highest perceivable rewards from among a predetermined set of actions. Such an automaton is referred to as a *Learning Automaton* (LA).

The field of LA was originally proposed by the astonishing work of Tsetlin in the early 1960's. The concept of the LA is an intuitive outcome resulting from the fusion of the research in modeling the observational behaviors of individuals and of the computational sciences. Narendra and Thathachar, the early pioneers in this field, have described the LA as a:

"... decision maker agent which operates in the random Environment and respectfully updates its policy for choosing actions based of the perceived (sequence of) responses. The decision making agent (or the automaton) in such an action-feedback configuration, is referred to as the learning automaton (LA). The automaton agent can choose among a set of predefined actions and correspondingly the Environment favors or opposes the action with a certain probability" [**?**].

Based on the algorithm used in designing the LA, we can categorize an LA to fall into one of two major classes:

- *Fixed Structure Stochastic Automata (FSSA)*: The algorithm used in this category of LA is based on a fixed, non-time-varying Markov Chain (MC).
- *Variable Structure Stochastic Automata (VSSA)*: In this class of LA, the action probabilities are given as a vector which is updated at every iteration of the algorithm by applying a predetermined update rule. This update rule can be either linear or non-linear.

For a comprehensive review of the families of FSSA, we refer the reader to Tsetlin's paper [**?**], and to the book by Narendra and Thathachar [**?**], as it has also explored these machines in fair detail. The authors of [**?**] have reviewed the first three decades of the research in the field of LA. It is worth noting that all the major efforts in the

contemporary LA research have focused on VSSA learning schemes which also includes the families of estimator algorithms [**?**], [**?**] discussed later. Although we will extensively work with all of them, we will omit an overview of all the above-mentioned families as it is found in [**?**].

As a consequence of this, LA, by definition, is an excellent candidate paradigm that can be applied to a spectrum of stochastic problems where an optimal action needs to be chosen from among a set of actions.

The computational modeling of the LA goes back to research works done by psychologists such as Atkinson [**?**] and Bush [**?**]. In the early 1960's and 1970's, Russian scientists such as Tsetlin, Krinsky and Krylov [**?**], and Varshavskii [**?**] extensively studied the various structures for the LA such as their deterministic and stochastic versions. These studies further motivated worldwide theoretical research and the corresponding experimental studies for the last four decades.

The field of stochastic learning, and specifically LA, encompasses a wide range of applications. These include solving network and communication problems [**?**], [**?**], and [**?**]. LA have also been applied successfully to address enterprise control problems such as network call admission [**?**], traffic control and quality of service routing [**?**] and Intelligent Vehicle Control [**?**] and [**?**]. In the more general field of AI, LA have been applied to Neural Network Adaptation [**?**], training Hidden Markov Models [**?**] and have been used to address the Distributed Scheduling problem [**?**] and in microaggregation [**?**]. A comprehensive review of the applications of LA is found in [**?**], [**?**], [**?**], [**?**] and [**?**], and in more recent survey journal issues in the field [**?**], [**?**].

2.3 The OMA

The OMA is a FSSA designed to solve the EPP. It is defined as a quintuple with R actions, each of which represents a specific class, and for every action there exist a fixed number of states, N. Every abstract object from the set $\mathcal{O}$ resides in a state identified by a state number, and it can move from one state to another, or migrate from one group[3] to another. Thus, if the abstract object O_i is located in state ξ_i belonging to a specific group (an action or class) α_k, we say the object O_i is assigned to the class number k.

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

We shall formalize the LA as follows: For every action α_k, there is a set of states $\{\phi_{k1}, \cdots, \phi_{kN}\}$, where N is the fixed depth of the memory, and where $1 \leq k \leq R$ represents the number of desired classes. We also assume that ϕ_{k1} is the most internal state and that ϕ_{kN} is the boundary state for the corresponding action. The response to the reward and penalty feedback are defined as follows:

- **Reward:** Given a pair of physical objects presented as a query $\langle A_i, A_j \rangle$, if both O_i,and O_j happen to be in the same class α_k, the reward scenario is enforced, and they are both moved one step toward the actions's most internal state, ϕ_{k1}. This is depicted in **?? (a)**.
- **Penalty:** If, however, they are in different classes, α_k and α_m, (i.e., O_i, is in state ξ_i where $\xi_i \in \{\phi_{k1}, \cdots, \phi_{kN}\}$ and O_j, is in state ξ_j where $\xi_j \in \{\phi_{m1}, \cdots, \phi_{mN}\}$) they are moved away from ϕ_{k1} and ϕ_{m1} as follows:
  a) If $\xi_i \neq \phi_{kN}$ and $\xi_j \neq \phi_{mN}$, then we move O_i and O_j one state toward ϕ_{kN} and ϕ_{mN}, respectively. This is pictorially given in **?? (b)**.

[3] To be consistent with the terminology of LA, we use the terms "action", "class" and "group" synonymously.

b) If $\xi_i = \phi_{kN}$ or $\xi_j = \phi_{mN}$ but not both (i.e., only one of these abstract objects is in the boundry state), the object which is not in the boundary state, say O_i, is moved towards *its* boundary state as shown in **?? (c)**. Simultaneously, the object that is in the boundary state, O_j, is moved to the boundary state of O_j. Since this reallocation will result in an excess of objects in α_k, we choose one of the objects in α_k (which is not accessed) and move it to the boundary state of α_m. In this case, we choose the object nearest to the boundary state of ξ_i, as shown in **?? (c)**.

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

The above rules are figuratively displayed in Figure **??**, and Algorithm **??** presents the algorithmic representation of the technique described above. It invokes the procedures "*ProcessReward*" and "*ProcessPenalty*" given in Algorithms **??** and **??** respectively.

The analysis of the OMA's convergence is not a trivial exercise since it is an interaction of the query generating system (the Environment) and the various objects. It can thus be interpreted as being a "cooperative" (as opposed to a "competitive" automaton game) **[?]**. From the experimental results, the authors of **[?]** claim that the scheme is ϵ-optimal. The experimental results obtained by simulating the OMA against benchmark Environments is given later, in Section **??**.

Algorithm 1 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 $\{1, 2, \cdots R \times N\}$, 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}                          ▷ This is described in the text
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:     Print out the partitions based on the states {ξ_i}
12: end
```

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

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.

begin
  if $\xi_i \bmod N \neq 1$ **then** ▷ Move O_i towards the internal state
    $\xi_i = \xi_i - 1$
  end if
  if $\xi_j \bmod N \neq 1$ **then** ▷ Move O_j towards the internal state
    $\xi_j = \xi_j - 1$
  end if
end

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

Input:

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

Output:

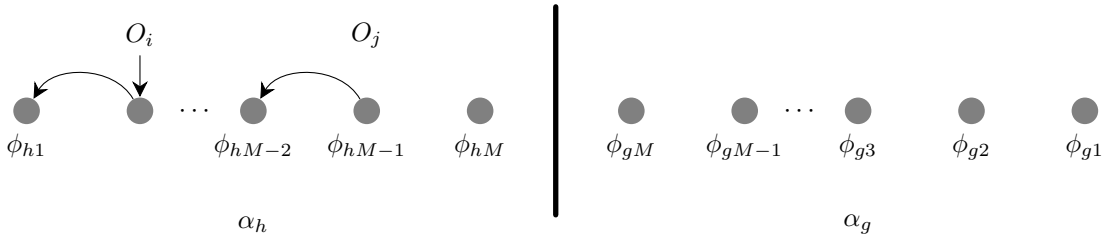
- The next states of the O_i's.

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

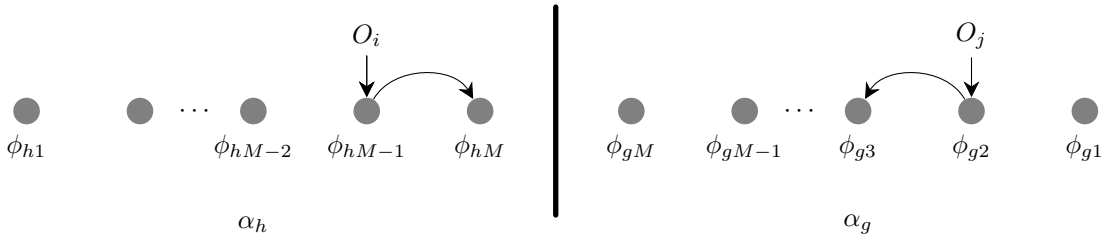
3 Overview of Our Solution

The fundamental problem that we now encounter is one of "filtering out" the noisy or divergent queries from a stream of queries generated by the Environment, $\mathbb{E}$. There are numerous filtering approaches available in literature. But considering the fact that the queries appear in "real-time", a suitable method must be simple and fast, and yet meaningful and flexible enough for it to be synthesized in conjunction with the OMA.

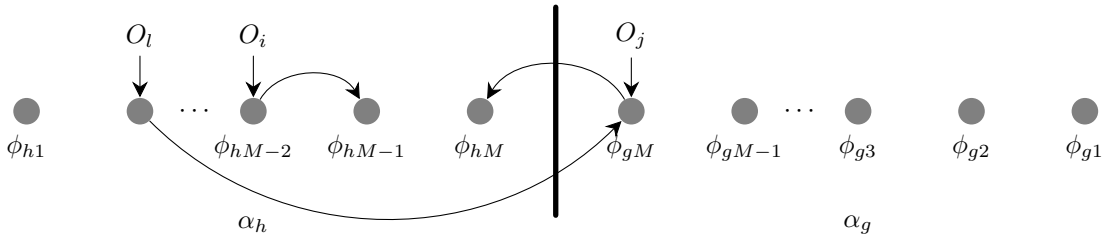
The first step is to develop a formal model for the Environment that generates the queries from which the partitioning is to be learned.



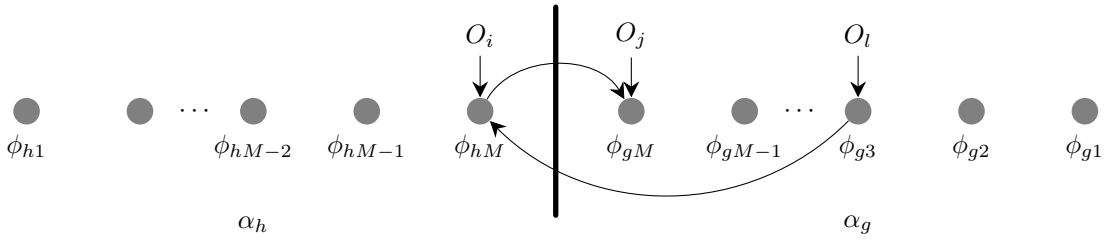
(a) On reward: Move the accessed abstract objects $\langle O_i, O_j \rangle$ towards the extreme states.



(b) On penalty: Move the accessed abstract objects $\langle O_i, O_j \rangle$ towards their boundary states (Case 1).



(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 (Case 2).



(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 (Case 3).

Figure 2: Transition rules for the two-action OMA. The details of these transitions are described in the text.

3.1 Modeling the Noise-less Environment

In an "un-noisy" Environment, i.e. in the absence of uncertainty, we can denote the actual value of this relation between A_i and A_j by a quantity $\mu^*_{i,j}$, expressed[4] as:

[4]The reader will observe that this is the *total* probability of $\mathbb{E}$ presenting the pair $\langle A_i, A_j \rangle$.

$$\begin{aligned}\mu^*_{i,j} = &\ P(R_k)\cdot P(A_j|A_i)\cdot P(A_i), \forall i,j \text{ if } \langle A_i, A_j\rangle \in R_K, \text{ with } k\in\{1,\cdots,R\},\\ &= 0 \text{ otherwise,}\end{aligned} \tag{1}$$

where $P(R_k)$ is the probability that the first element, A_i, is chosen from the group R_k, and $P(A_j|A_i)$ is the conditional probability of choosing A_j, which is also from R_k, after A_i has been chosen. We shall use Equation (**??**) to represent the elements of the overall policy matrix as described below.

Since $\mathbb{E}$ chooses the elements of the pairs from the other groups uniformly, the matrix $\mathcal{M}^* = \left[\mu^*_{i,j}\right]$ is a *block-diagonal* matrix[5] of the form:

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}^*_1 & \underline{0} & \cdots & \underline{0} \\ \underline{0} & \mathcal{M}^*_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{0} & \cdots & \cdots & \mathcal{M}^*_R \end{bmatrix} \tag{2}$$

where $\underline{0}$ represents a square matrix containing only 0's. In the interest of simplicity, we shall assume that the relevant distributions are uniform[6]. In other words, if the first element specified by $\mathbb{E}$ is A_i from class G_r, the probability of the second element being any of the other elements in G_r is equally divided between these elements. Similarly, the probability of the second element in the pair being any of the other elements *not* in G_r, is also assumed to be divided equally between them. We now derive the explicit form for the each of $\mathcal{M}^*_i$s.

Theorem 1. *The matrix $\mathcal{M}^*_r, (1\le r\le R)$, is a matrix of probabilities of size $\frac{W}{R}\times\frac{W}{R}$ possessing the following form:*

$$\mathcal{M}^*_r = \begin{bmatrix} 0 & \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} \\ \frac{R}{W(W-R)} & 0 & \cdots & \frac{R}{W(W-R)} \\ \vdots & & \ddots & \vdots \\ \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} & 0 \end{bmatrix} \tag{3}$$

Proof. We shall show the result for the matrix $\mathcal{M}^*_1$, whence the proof for the general $\mathcal{M}^*_r$ would be obvious.

$\mathcal{M}^*_1$ is a $\frac{W}{R}\times\frac{W}{R}$ matrix whose element $[i,j]$ represents the probability of $\langle A_i, A_j\rangle$ being accessed simultaneously where $1\le i,j\le\frac{W}{R}$. Clearly, since the pair $\langle A_i, A_i\rangle$ cannot be presented by $\mathbb{E}$, the diagonal elements are all 0.

Consider now the scenario when the first element chosen by $\mathbb{E}$ is A_1. Then the second element can be any one of the A_j's, $2\le j\le\frac{W}{R}$, and hence j can take $\frac{W}{R}-1$ possible values. Since all of these elements are equally likely, the conditional probability of j given that $i=1$ equals $\frac{1}{\frac{W}{R}-1}=\frac{R}{W-R}$. Since the total probability $P(u,v)=P(u|v).P(v)$, and since $P(A_1)=\frac{1}{W}$, the total probability $P(A_1,A_j)=\frac{R}{W(W-R)}$, proving the explicit form for the first row of $\mathcal{M}^*_1$. The expressions for the other rows in $\mathcal{M}^*_1$ follow in an analogous manner.

A simple algebraic exercise will demonstrate that the sum of all the elements in $\mathcal{M}^*_1$ is $\frac{1}{R}$.

A similar argument can be used to show that the contents of any $\mathcal{M}^*_r$ obeys Equation (**??**), and that the sum of all the probabilities in $\mathcal{M}^*$, given in Equation (**??**), is unity. Hence the result! □

[5]One might argue that the order of the indices may not necessarily result in a block matrix, but rather in a sparse matrix. However, it can be easily seen that by an appropriate re-mapping of the indices, in which the elements of the partitioning Ω^* are adjacent, one can obtain such a block matrix.

[6]Extending these results for a non-uniform setting is rather trivial.

We now examine the real-world scenario, i.e., where the Environment is noisy.

3.2 Modeling the Noisy Environment

In a real-world scenario where the Environment is noisy, i.e., the objects from the different groups can be paired together in a query, the general form for $\mathcal{M}^*$ is:

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}_1^* & \underline{\theta} & \cdots & \underline{\theta} \\ \underline{\theta} & \mathcal{M}_2^* & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{\theta} & \cdots & \ldots & \mathcal{M}_R^* \end{bmatrix}, \tag{4}$$

where $\underline{\theta}$ and $\mathcal{M}_r^*$s are specified as per Equations (**??**) and (**??**) respectively.

Theorem 2. *In the presence of noise in $\mathbb{E}$, the entries of the pair $\langle A_i, A_j \rangle$ can be selected from two different distinct classes, and hence the matrix specifying the probabilities of the accesses of the pairs obeys Equation (**??**), where:*

$$\underline{\theta} = \theta_o \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & 1 \end{bmatrix}, \tag{5}$$

and

$$\mathcal{M}_r^* = \theta_d \cdot \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix}, \tag{6}$$

*where, $0 < \theta_d < 1$ is the coefficient which specifies the accuracy of $\mathbb{E}$, and θ_o is related to θ_d as per Equation (**??**).*

Proof. As in the case of Theorem **??**, we show the result for the matrix $\mathcal{M}_1^*$, whence the proof of the general $\mathcal{M}_r^*$ can be trivially obtained.

$\mathcal{M}_1^*$ is a $\frac{W}{R} \times \frac{W}{R}$ matrix whose $[i, j]$ entry represents the probability of $\langle A_i, A_j \rangle$ being presented by $\mathbb{E}$. If $\mathbb{E}$ chooses the first entry to be A_1, it can choose the second element from the same class with a probability θ, and an element can be chosen from any of the remaining classes[7] with probability $1 - \theta$. Since $\mathcal{M}_1^*$ is symmetric, all the entries along the diagonal are zero and the off-diagonal entries represent the within-class probabilities of $\langle A_i, A_j \rangle$ where $2 \leq j \leq \frac{W}{R}$. Since all the $[1, j]$ entries of $\mathcal{M}_1^*$ are equally likely, the form of Equation (**??**) is clear, where θ_d is the total probability of any of the elements, other than A_1, in $\mathcal{M}_1^*$ being chosen.

Using the same analogy, we can factor out the equal elements of $\mathcal{M}_r^*$ (i.e., that all the non-diagonal entries are equal and that the diagonal elements are all 0), to represent each block as Equation (**??**).

If A_2 belongs to a class distinct from A_1, as opposed to the case of Theorem **??**, the $\underline{\theta}$ matrices in the same block-row with $\mathcal{M}_1^*$ of $\mathcal{M}^*$, cannot be zero matrices anymore. Thus, each entry in the first row outside of $\mathcal{M}_1^*$ must be

[7] This must be contrasted with Theorem **??** where $\mathbb{E}$ cannot choose the second element from any other class.

set to the probability value of A_j being selected ($\frac{W}{R}+1 \leq j \leq W$) from any other class. Since all the other classes are equi-probable in being selected, we need to only determine this probability for any one element and see that the rest of the entries possess the same values. We let this probability, the element $[1, j], \frac{W}{R}+1 \leq j \leq W$ of $\underline{\theta}$, be θ_o.

Since we have a total of W elements in every row of $\mathcal{M}^*$, and since there are $\frac{W}{R}$ of them in each matrix $\underline{\theta}$, in any of the rows of $\mathcal{M}_1^*$ there will be $W - \frac{W}{R}$ elements which will all equal to θ_o. We can divide the rest of the remaining elements into $R-1$ groups of size $\frac{W}{R}$ and use Equation (**??**) to yield a block matrix representation.

To obtain the value of θ_o, we use the fact that the summation of all the elements in $\mathcal{M}^*$ must be equal to unity. Since we have W rows with identical summations over every rows, the sum of every row must be equal to $\frac{1}{W}$. Thus, with a simple algebraic computation, we can derive the values for the sum of the first row to be:

$$\theta_d(\frac{W}{R}-1) + \theta_o(W - \frac{W}{R}) = 1, \tag{7}$$

whence, we can see that θ_d has the form:

$$\theta_d = \frac{1 - \theta_o(W - \frac{W}{R})}{\frac{W}{R} - 1}. \tag{8}$$

Hence the theorem! □

4 Pursuit Concept

We shall now present one such filtering approach that uses the Pursuit paradigm from the field of LA. Traditional LA work with the understanding that the actions are chosen purely based on the "state" in which the machine is. Thus, when the LA receives the feedback from $\mathbb{E}$, it invokes its state update function to force the machine to go to a new state. This new state is then used by the output function to determine its next action. The reader will observe that this is exactly the paradigm followed by both the OMA and its enhanced version. The state update function need not be deterministic. Rather, it could also be stochastic and this concept has been used to further enhance the OMA to create the SMA [**?**].

While the above describes the FSSA strategy of designing LA, the VSSA schemes do not specifically work with "states". Rather, all the state information is encapsulated within an action probability vector. Thus, the feedback from the Environment forces the LA to change its action probability vector using which it chooses the next action.

Both of these strategies seem to completely ignore any estimation of $\mathbb{E}$'s reward/penalty probabilities. To take these into consideration, Estimator/Pursuit LA utilize "cheap" estimates of the $\mathbb{E}$'s reward probabilities to make them converge by an order of magnitude faster. This concept is quite simply the following: Inexpensive estimates[8] of the reward probabilities can be used to *rank* the actions. Thereafter, when the action-probability vector has to be updated, it is done not on the basis of the Environments's response alone, but also based on the ranking of these estimates. Thus, as the estimates becomes more accurate, the superior actions will be chosen more often, and the LA will converge to them at a much faster rate.

An estimation scheme can be viewed as the filtering approach. Once the LA has computed the estimates, it is capable of filtering out the less informative responses from $\mathbb{E}$, and to thus enhance the choice of its superior actions. This is, precisely, what we will now do to improve the speed of the OMA by an order of magnitude.

[8] Of course, these estimates can be obtained using either a Maximum Likelihood or a Bayesian paradigm.

To achieve this goal, we shall invoke the Pursuit principle to use estimates to efficiently approximate the *querying* system. The traditional OMA learns the best partitioning by keeping track of the abstract objects that are jointly accessed. It ignores though, the estimates of the frequencies with which the pairs are queried.

Obviously, we could obtain a rough estimate of how frequently two objects can appear in pairs, after a few iterations of the OMA. Our goal is to utilize this information to evaluate (or, one can say "estimate") the "degree" of certainty of a query pair. Our strategy is the following: If the estimated certainty is less than a pre-defined threshold, we opt to treat the query in question as a divergent query – in the spirit of the pursuit paradigm.

4.1 Extracting Convergent Information from $\mathbb{E}$.

In general, a large amount of uncertainty in partitioning can severely degrade the performance of the OMA or even distract the OMA from the correct partitioning. It also reduces the convergence rate both in the initial and final phases. One can visualize the divergent queries as those causing "backward" steps from the OMA's final equilibrium state. Although the authors of [**?**] improved the algorithm in various ways, the divergent queries provided by the Environment, $\mathbb{E}$, still have a detrimental effect on the OMA's convergence rate and performance.

4.2 How the POMA is Designed

We have now set the framework by which the pursuit concept can be incorporated into the OMA. The foundations, of course, rely on the models of the Environment for noise-free and noisy queries presented in Sections **??** and **??** respectively. What we intend to do is to process the set of incoming queries and to reject those that cause the traditional OMA to linger in inaccurate configurations. We shall accomplish this by resorting to an estimation strategy.

We motivate our discussion by considering a simple method for estimating the values of the accuracy of the pairs in every query, and for using these estimates as a *policy* to accept/reject the queries in question. Consider a typical partitioning problem with W objects and R classes. As we know, the classification into groups is achieved by observing the objects in the sequence of queries provided by $\mathbb{E}$. Since the queries come in pairs of the form $\langle A_i, A_j \rangle$, a natural way to approximate the true value of this bilateral relationship between A_i and A_j is by the virtue of the mean reward received when the objects are paired (i.e., given by $\mathbb{E}$) together.

In a real world scenario, since the $\mathbb{E}$'s true statistical model is unknown, the expressions given by Equations (**??**) and (**??**) can only be estimated empirically through observing a set of queries. In the presence of noise though, we need to devise a measurable quantity which makes the algorithm capable of recognizing divergent pairs.

Whenever one desires to resort to estimation in a real-life problem, he has to decide on the *quantity* to be estimated, and then on the method by which the estimation is accomplished. In this case, since the information about the convergent/divergent queries resides in the matrix $\mathcal{M}^*$ defined in Theorems **??** and **??**, our task is to estimate the corresponding entries in $\mathcal{M}_i^*$ in Equation (**??**). On the other hand, the *method* of estimation that we will invoke will be the traditional Maximum Likelihood (ML) method where any quantity is estimated by the ratio of the number of times that the corresponding event occurs to the total number of trials.

Observe that whenever a real query $\langle A_i, A_j \rangle$ appears, we will be able to obtain a simple ML estimate of how frequently A_i and A_j are accessed concurrently. Clearly, by virtue of the Law of Large Numbers, these underlying estimates will converge to the corresponding probabilities of $\mathbb{E}$ actually containing the elements A_i and A_j in

the same group. As the number of queries processed become larger, the quantities inside $\mathcal{M}_i^*$ will become significantly larger than the quantities in each of the $\underline{\theta}$ matrices. An example of how the estimate of $\mathcal{M}^*$ looks is displayed in Figure **??** for the simple case when we have three block matrices, i.e., when we are dealing with three distinct partitions. From the figure, one will observe that the estimates corresponding to the matrix $\mathcal{M}_i^*$ have much higher values than the off-diagonal entries. This immediately informs us of the fact that these off-diagonal entries represent divergent queries which will tend to move the objects away from their accurate partitions.

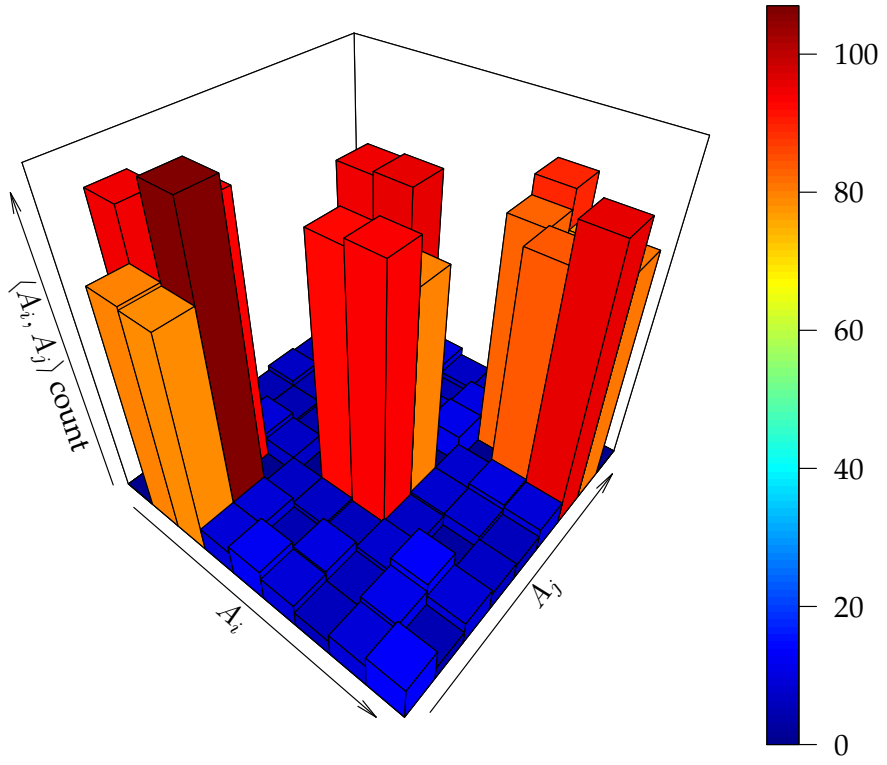


Figure 3: The estimation of $\mathcal{M}^*$ which allows us to invoke the pursuit concept. The number of partitions, R, in this case, is 3 and $p = 0.8$.

Intuitively, the pursuit concept for the OPP can be best presented by a matrix of size $W \times W$ where every entry will capture the same statistical measure about the stream of the input pairs. For the sake of simplicity, we use a simple averaging and denote this matrix by $\mathcal{P}$. This is depicted[9] in Figure **??**. Every block represents a pair and the height of the block is set to the frequency count of the reciprocal pair. To obtain the average frequency of each pair we let the OMA iterate for a sufficient time, say k iterations, and at every instant we update the value of the matrix $\mathcal{P}$ respectively. In this way, at the end of the k-th iteration, we have simply estimated the frequency of each pair. At this point, by observing the values of the matrix, the operator can find an appropriate threshold ($\tau > 0$)

[9]Please note that the matrix can be scrambled and it is rather trivial to group the elements into blocks as it is shown in the picture.

to be adopted as the accept or reject policy for any future occurrence of this particular pair of objects. If we let the algorithm to collect large enough number of pairs, in other words as $t \to \infty$, by the law of large numbers we can claim that:

1. $\exists \theta^* \mid \forall i, j : \underline{\theta}_{i,j} < \theta^*$.
2. $\forall\ i, j :\ \ \mu_{i,j} \gg \theta^*$.

The above observation leads us to a very simple conclusion. If we utilize a user-defined threshold, τ, (which is reasonably close to 0), after the estimation has converged, we will be able to compare every estimate to τ, and make a meaningful decision about the identity of the query. If the corresponding estimate is less than τ we can confidently assert that it came from a divergent query. In other words, by merely comparing the estimate to τ we can determine whether a query pair $\langle A_i, A_j \rangle$ should be processed, or quite simply, be ignored. This leads us to algorithm POMA below. Before the estimate has converged, we must merely invoke the OMA whenever a query is processed. However, after convergence, the identity of every query is evaluated, and every query which is inferred to be divergent is ignored. Further, if the estimate is greater than τ, the pair is used to invoke the Reward and Penalty functions of the original OMA algorithm given in Algorithms **??** and **??** respectively.

The question of when the estimate has converged is decided by simply assuming that a reasonable convergence has occurred by a certain number of iterations, say κ. Thus, after κ iterations, if $\mathcal{P}\left[A_i, A_j\right] > \tau$ then the pair is considered as a valid pair (converging pair), and otherwise, it would be an outlier (diverging pair) which should be filtered out. After receiving each pair, we update the statistics to reflect the newly gained information. The rest of the algorithm is analogous to the OMA. This concept is demonstrated algorithmically in the POMA algorithm, and presented formally in Algorithm **??**.

5 Experimental Results

Now that we have discussed the issues concerning incorporating the theory of the pursuit paradigm into the OMA, we will report the results concerning the data generation and the POMA's convergence.

5.1 Data Generation

The approach which is used for generating the query pairs is identical to what was discussed earlier in Section **??**. As before, there are W objects, and R groups, and the Environment , $\mathbb{E}$, is characterized by a probability p with which pairs of objects from the same group are queried together.

To be specific, the first object of each query, is drawn randomly and uniformly from among the W available objects. Thus, the probability of selecting any given object is $\frac{1}{W}$. In this case all the R groups are equally-likely, and will be selected with a probability of $\frac{1}{R}$.

There are now two scenarios for choosing the second element:

- **The second element belongs to the same group**: In this scenario, we choose one from among the $W - 1$ remaining objects. But to ensure that it belongs to the same class as the first-selected object, we toss a coin and choose to select the same class with the Environments's probability, p. If the random number is less than

Algorithm 4 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.

begin
  The initialization of $\{\xi_p\}$, as described in the text.
  for a sequence of $i = 1, \cdots, T$ queries **do**
    Read query $\langle A_i, A_j \rangle$
    $\mathcal{Z}[A_i, A_j] = \mathcal{Z}[A_i, A_j] + 1$
    $\mathcal{Z}[A_j, A_i] = \mathcal{Z}[A_i, A_j]$
    $\mathcal{P}[A_i, A_j] = \frac{\mathcal{Z}[A_i, A_j]}{\sum_{k,l=1}^{W} Z[A_k, A_l]}$ ▷ Update the statistics
    if $i < \kappa$ **then** ▷ Statistics have not converged; Invoke OMA
      if ξ_i div $N = \xi_j$ div N **then** ▷ The partitioning is rewarded
        Call ProcessReward($\{\xi_p\}, A_i, A_j$)
      else ▷ The partitioning is penalized
        Call ProcessPenalty($\{\xi_p\}, A_i, A_j$)
      end if
    else if $\mathcal{P}[A_i, A_j] \geq \tau$ **then** ▷ Statistics have converged
      if ξ_i div $N = \xi_j$ div N **then** ▷ Reward partitioning if $\mathcal{P}[A_i, A_j] \geq \tau$
        Call ProcessReward($\{\xi_p\}, A_i, A_j$)
      else ▷ Penalize partitioning if $\mathcal{P}[A_i, A_j] \geq \tau$
        Call ProcessPenalty($\{\xi_p\}, A_i, A_j$)
      end if
    else ▷ Filter the Divergent queries
    end if
  end for
  Print out the partitions based on the states $\{\xi_i\}$
end

p, we choose the second object from among the $\frac{W}{R} - 1$ objects in the same class. Thus, the probability of the second object being chosen from the same group is $\frac{p}{\frac{W}{R} - 1}$.

- **The second element belongs to a different group**: In this case we have to choose the second object from one of the $R - 1$ remaining groups. This occurs with the probability $1 - p$. Since we choose the other object in an equally-likely manner, the probability of any object being chosen from a different group is $\frac{1-p}{W - \frac{W}{R}}$.

Indeed, the above-mentioned arguments constitute another form of interpreting Equation (**??**). Both of these methods lead to identical partitioning scenarios. Using this technique, $\mathbb{E}$ is capable of generating a continuous stream of query pairs while maintaining its probabilistic integrity and the system's overall stability.

The hope is that by generating the query pairs in this manner, and processing them using the POMA, the final partitioning will lead to an identical structure, as depicted in **??**.

5.2 Setting the Threshold Value

As discussed in Section **??**, the threshold, τ, is a user-defined parameter which controls whether a certain query is to be considered as a divergent query or not. There are two major factors affecting the value of τ. We discuss each of these below.

The reader must understand that the value of τ is very crucial for the POMA to not only converge but to even operate in a meaningful manner. The first issue to be discussed is that of knowing when the estimates themselves converge. When the number of queries is small, none of the estimates will be accurate. In such a case, it is meaningless to try to infer when a query is divergent. This scenario must be accounted for. Secondly, if the value of τ is set too high, none of the queries will be processed because the estimate of that query occurring may not exceed the threshold. On the other hand, if the value is set too low, i.e. $\tau \approx 0$, all the queries will be considered to be non-divergent queries and the operation of the POMA will be identical to that of the original OMA. This too is an issue that we have to discuss in greater detail.

- **Specifying the process in the initial phase**: In the initial phase of the algorithm, the estimates for the queries are unavailable. Thus, since we cannot infer the divergent queries, it only makes sense to consider every single query and to process them using the OMA's Reward and Penalty functions. That being said, these are not futile operations. The learning will be at least as effective as the OMA. On the other hand, the occurrence of these queries permits us with the opportunity to estimate the query statistics. Since the objects in each class are equally-likely to happen and the classes are equi-probable, after about $\left[\left(\frac{W}{R}\right)^2 - \frac{W}{R} \right] \times R$ iterations, it is very likely that at least one of the objects at each class has been included in the stream of k queries received by the POMA so far. This value, $k \geq \left[\left(\frac{W}{R}\right)^2 - \frac{W}{R} \right] \times R$, can be considered as the lower-bound for the number of iterations needed for any meaningful initialization, and is thus set to be the value of κ[10].

- **Setting the value of** τ: The difficult problem that the user has to resolve involves that of setting the value of τ. The reader will recall from Theorem **??** that the entries of the off-diagonal matrices, in the noise-free case, are all zero. Thus, in a noise-free Environment, we are guaranteed to only receive query pairs from the same group. The complexity arises when we are dealing with noisy Environments. In these cases, from Theorem **??**, we understand that there is a non-zero probability of having queries presented from different groups. From the form of the $\underline{\theta}$ matrices, we know that the entries they contain are all unity multiplied by a constant θ_o as per Equation (**??**). As explained in [**?**], it can be shown that a tighter bound for τ is $0 < \tau < \frac{1}{W^2 - W}$.

  This now gives us a methodology by which we can filter out the divergent queries. Indeed, if we are given an unending query stream, and we estimate the probabilities of objects appearing from different groups, the estimates will converge to θ_o if the model satisfies the condition of Theorem **??**. Thus, if we succeed in setting the threshold value τ to be marginally greater than θ_o, after the estimation has converged, we will be able to transform the queries in the noisy Environment to appear as if they come from a noise-free Environment by filtering out the queries whose access probabilities are less than θ_o.

  This leads us to a simple mechanism by which the threshold τ can be set. As the queries appear, we will observe that the entries within the same groups will be quite high, i.e., comparable to the quantity defined as θ_d in Equation (**??**). Thus, if the conditions of Theorem **??** are satisfied, the estimates of the entries will all be

[10] In some cases, in the experimental results reported in **??**, we have set the values of κ and τ to alternate values that are specified in the table, so as to enhance the convergence of the algorithm.

in the neighbourhood of θ_d or in the neighborhood of θ_o. After a reasonable number of queries are processed a simple search will enable us to determine a value of τ which is marginally larger than the largest of the off-group elements.

The number of queries to be processed depends on how noisy the Environment is – we need to make sure that the off-diagonal elements are seen minimally.

- It is trivial for us to see that the value of κ should depend on the difficulty of the problem to be solved, i.e., on the complexity of the Environment and the number of possible partitions. In other words, depending on the difficulty of the Environment, the size of the set to be partitioned and the number of classes, one will have to increase the value of κ or adjust the τ accordingly. The actual values used in our experiments are specified in the tables below.

5.3 Simulation Results for the OMA

Experimentally, the works of earlier researchers compared the BAM and the OMA for various values of W and R. The results, which are also reviewed here, use the number of states per action to be 10. The convergence criteria utilized here is identical to the one used in [**?**]. The OMA does not merely improve the convergence rate significantly over its competitors; it is also computationally less expensive. For further additional details and more comparative results involving the OMA and the BAM, we refer the reader to [**?**]. However, since we will be using the OMA as a benchmark, it is prudent to explain the Environment and the settings in more detail. This is done below.

While the OMA algorithm is iterating towards Ω^+, there will, hopefully, be an iteration index at which juncture all the objects reside in an accurate partitioning. The number of iterations that have taken place up to this point, which relates to the first time when the OMA has reached the correct partitioning, is recorded. Additionally, we have also recorded the iteration index when all the objects move into the most internal state of their respective classes. Both of these time instances indicate the convergence of the OMA. To be more specific, consider the case when it takes the OMA 200 iterations to settle into the final configuration. However, it may first reach to this correct classification at iteration 110. In such a case, the algorithm would report the pair $\langle 110, 200 \rangle$ as the OMA's convergence results.

The experimental results[11] for the OMA are given in Table **??**. In the table, the probability characterizing $\mathbb{E}$ varies between 0.7 to 0.9. Clearly, as the value of p increases, the queries are more informative, and the convergence occurs at a faster rate. For example, we see in Table **??** that for the case where $R = 3$ and $W = 9$, and when the probability of receiving a paired query was $p = 0.9$, it took the OMA, on average, 44 iterations to fall into a correct partitioning for the first time. Further, it took 110 iterations, on average, to fully converge. All the simulations reported were done on an ensemble of 100 experiments to guarantee statistically stable results.

The convergence of the OMA with regard to time is also very interesting. This is depicted in Figure **??** for the case where $W = 9$ and $R = 3$. This figure presents the number of objects that are not correctly placed with regard

[11] In our current implementation, we have compared the original OMA algorithm described in [**?**] with our own simulation results. These are the results shown in Table **??**. In our "improved" implementation, whenever two objects are to be switched from their boundary states, we decide on the one to be moved by randomly tossing a coin. These "improved" results, obtained after averaging over 100 independent experiments, are superior to those given in [**?**].

Table 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.

to the true underlying partition at any given time instant. The reader will observe that the OMA starts with a large number of objects in the wrong partition, and steadily decreases this index to a very small value. The behavior, clearly, is not monotonic for a single experiment. However, if one takes the ensemble average of this metric over a large enough number of experiments, the performance is much more monotonic in behavior. This is clear from Figure **??**.

5.4 Simulation Results for the POMA

In order to evaluate the effectiveness and convergence speed of the POMA, we have compared our results with those presented in [**?**] and those reported in Table **??** for various values of R and W. In our experiments, the number of states in every action was set to 10, and the convergence was expected to have taken place as soon as all the objects in the POMA fell within the last two internal states[12]. The approach explained in Section **??** was followed quite precisely with regard to setting the parameters of the POMA. The results obtained are quite amazing and summarized in Table **??**. In the experiments, the default entry in the second-last column for τ is $\tau^* = \frac{1}{W^2}$, and otherwise the value of τ is specified in each row. Similarly, the default entry in the last column for κ is $\kappa^* = R\left[(\frac{W}{R})^2 - \frac{W}{R}\right]$, and otherwise the value of κ is specified in each row. The simulation results are based on an ensemble of 100 runs with different uncertainty values, (i.e., values of p) ranging from $0.7 - 0.9$.

[12]One should remember that the estimation of the queries' probabilities must be achieved after every single query is received.

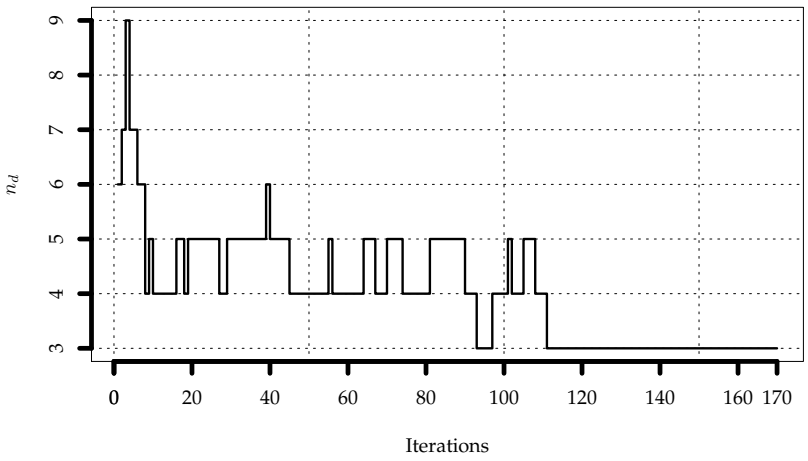


Figure 4: A plot of n_d, the number of objects in groups different from their true underlying partitions for the OMA, as the number of iterations (i.e. query pairs) proceed in a single run. Here, $W = 9$ and $R = 3$.

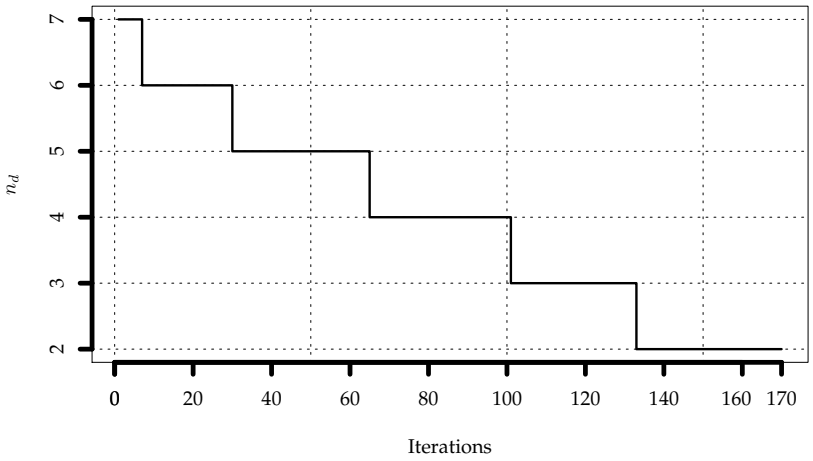


Figure 5: A plot of n_d, the number of objects in groups different from their true underlying partitions for the OMA, as the number of iterations (i.e. query pairs) proceed in an ensemble of 100 runs. Here, $W = 9$ and $R = 3$.

When we observe the execution of the algorithm, we observe that the argument presented in Theorem **??** becomes obvious as the total number of divergent query pairs is always much less than the number of convergent pairs. Thus, without loss of generality, the value of τ obtained by the steps described in the previous section, can be set as a *threshold* value for the POMA's accept/reject policy. Indeed, for a pair given by $\mathbb{E}$ at time t, if the corresponding estimate for the query has a value less than τ, the query is treated as a divergent pair. Otherwise the pair is considered to contain valuable information, and the POMA's Reward/Penalty functions are invoked. A comparative discussion of the POMA with the OMA is given in the next section.

Table 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.

6 Discussion

The incorporation of the pursuit phenomena into the traditional OMA enables the corresponding LA to gracefully, and yet effectively, dampen the diverging flow of the OMA's execution. Comparing Tables **??** and **??**, we observe that although the performance gain is insignificant in cases involving partitioning data sets of small-sizes and for simple Environments, the POMA significantly surpasses the OMA in difficult Environments and for complex partitioning problems. In this context we mention that a partitioning becomes increasingly challenging if the Environment is difficult to learn (i.e. it possesses a large amount of noise and thus has a large number of divergent

queries), or a large number of objects that are in each group, which also leads to a slow convergence rate. This is also obvious in Tables **??** and **??**.

To observe the efficiency of the POMA and to obtain a comparison with the original OMA proposed by Oommen *et al.* **[?]**, consider an easy-to-learn Environment of 3 groups with 5 objects in each group and where $p = 0.9$. It took the OMA $2,087$ iterations to converge. As opposed to this, the POMA converged in only $1,879$ iterations. On the other hand, given a difficult-to-learn Environment where $p = 0.9$, with 18 objects in 6 groups, the OMA needs 839 iterations to converge. The POMA required only 494 iterations to converge, which is nearly *two* times better than the original OMA. The algorithmic time complexity does not change either because the POMA involves only a *single* additional estimation and comparison operation in every iteration.

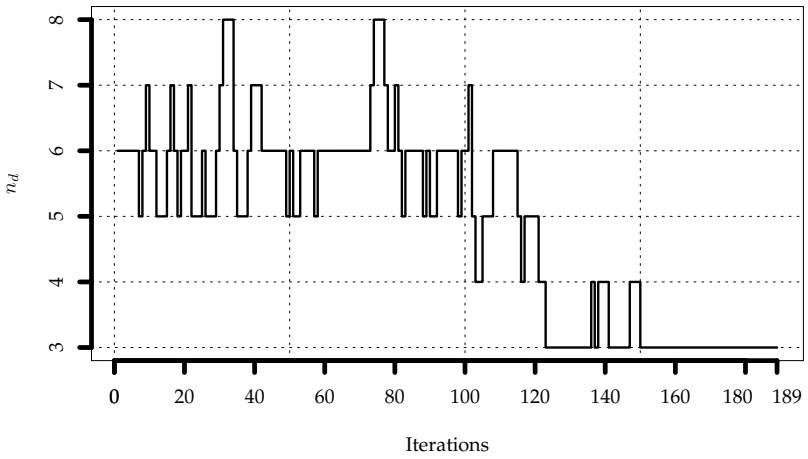


Figure 6: A plot of n_d, the number of objects in groups different from their true underlying partitions for the POMA, as the number of iterations (i.e. query pairs) proceed in a single run. Here, $W = 9$ and $R = 3$.

The convergence of the OMA with regard to time is also very interesting. This is depicted in **??** for the case where $W = 9$ and $R = 3$. This figure presents the number of objects that are not correctly placed with regard to the true underlying partition at any given time instant. The reader will observe that the POMA too starts with a large number of objects in the wrong partition, and steadily decreases this index to a very small value. The behavior, clearly, is not monotonic for a single experiment. However, if one takes the ensemble average of this metric over an ensemble of experiments the performance is more monotonic in behavior, as seen from **??**.

It is worth noting that the results of **??** does not take into account any of the improvements proposed by the authors of **[?]**. By merely using the OMA and the pursuit information, we can, indeed, achieve superior results. The advantage gained by incorporating the enhancements due to Gale *et al.* can also be used in conjunction with the pursuit paradigm to yield even better results, and this is the focus of our further research.

There is an alternate method to increase the accuracy of the OMA and this consists of increasing the number (to possibly "infinite") of states it possesses in each group. The difference between considering such an "infinite" number of states in the traditional OMA, and the way by which the POMA works, lies within the fact that the POMA operates in a higher dimension, utilizing statistical measures extracted from the input stream, and including them in the reinforcement policy. This cannot be achieved by merely resorting to the former scheme. Indeed,

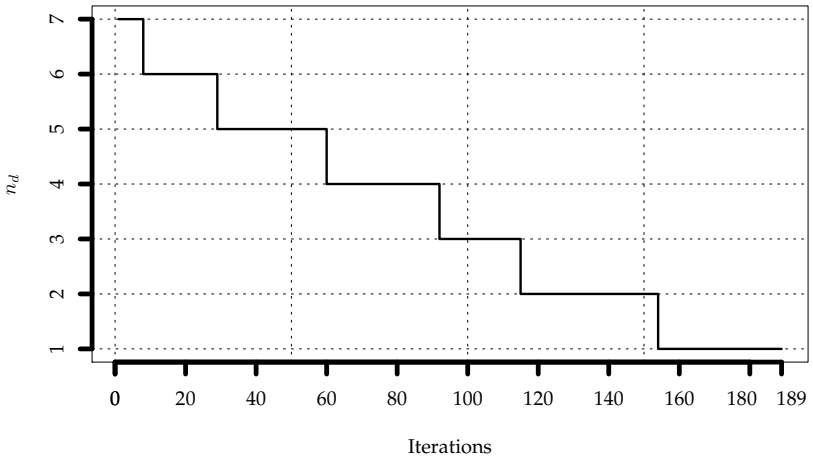


Figure 7: A plot of n_d, the number of objects in groups different from their true underlying partitions for the POMA, as the number of iterations (i.e. query pairs) proceed in an ensemble of 100 runs. Here, $W = 9$ and $R = 3$.

the statistical properties of the Environment is entirely neglected in the traditional OMA algorithm.

One can also make some interesting observations about the overhead computations required by the POMA. We would argue that since the version of the POMA that we have presented uses only simple averaging in its policy, in conjunction with the pursuit matrix, the computational performance is identical to that of the OMA. In spite of this, we obtain results that are *many* times superior. In our opinion, this is quite remarkable.

7 Conclusion

In this research, we introduced the Object Partitioning Problem (OPP), and we studied a special case where the number of objects in each group are equal. In the literature, this problem is referred to as the Equi-Partitioning Problem (EPP). We further reviewed the previously-proposed solutions for the OPP and the EPP. With regard to the EPP, we discussed, in detail, the Object Migration Automata (OMA). The latter is an LA-based approach to resolve the EPP. This solution has served as a benchmark for about three decades.

With regard to generating the query stream, we formally proposed the model for the Environment for the case when it was either noisy or noiseless. We argued that the so-called "divergent" queries can drastically slow down the convergence rate of the OMA. Consequently, we proposed that the Pursuit concept can be incorporated into and used as an effective accept/reject policy, which, in turn, led to a significant increase in the performance of the OMA. This led to the Pursuit OMA (POMA).

We also discussed how one could obtain a fair estimate for the POMA's threshold value, τ. With regard to demonstrating the strength of our new techniques, we also presented the experimental results obtained, on benchmark environments, that compared the POMA and the OMA. We were able to show how the POMA outperformed the OMA. As an example, in the partitioning of 18 objects into 6 groups, in an difficult-to-learn Environment, when the probability of receiving a divergent query was $p = 0.7$, the POMA performed nearly two

times better than the OMA. Indeed, in certain extreme environments it is almost *ten* times faster than the original OMA reported in the legacy papers. It is fascinating to note that the reduction in the number of iterations in the POMA can be seen to be a consequence of a very simple filtering phase, and this is valid in both simple/difficult Environments, and in both easy and hard-to-partition problems.

In summary, the introduced modification enhances the OMA's performance without any significant computational overhead in both cases, when the Environment is difficult to learn and the partitioning problem is inherently challenging.

In terms of future research, we are currently investigating how we can incorporate the Pursuit paradigm to mitigate certain deadlock situations that could be encountered in the OMA. We are also studying how variants of the OMA can be designed and implemented for unsolved problems such as the Unbalanced Partitioning Problem and the Proportional Partitioning Problem. Finally, we are considering the implementation of these methods to the problem of mapping processes onto processors in a distributed environment.

References

[1] R. C. Atkinson, G. H. Bower, and E. J. Crothers. *Introduction to mathematical learning theory.* Wiley, 1965.

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

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

[4] R. R. Bush and F. Mosteller. *Stochastic models for learning.* John Wiley & Sons, Inc., 1955.

[5] D. Ciu and Y. Ma. *Object Partitioning by Using Learning Automata*. PhD thesis, Carleton University, 1986.

[6] Eugene C. Freuder. The object partition problem. *Vision Flash*, -(4), 1971.

[7] E. Fayyoumi, and B. J. Oommen. Achieving Micro-Aggregation for Secure Statistical Databases Using Fixed Structure Partitioning-Based Learning Automata. *Systems Man and cybernetics, IEEE Transactions on*, 39(5):1192- 1205, 2009.

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

[9] M. Hammer and A. Chan. Index selection in a self-adaptive data base management system. In *Proceedings of the 1976 ACM SIGMOD international conference on Management of data*, pages 1–8. ACM, 1976.

[10] M. Hammer and B. Niamir. A heuristic approach to attribute partitioning. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*, pages 93–101. ACM, 1979.

[11] G. Horn. *An Interdisciplinary Approach to Optimisation in Parallel Computing*. PhD thesis, UNIVERSITY OF OSLO, 2012.

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

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

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

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

[16] K. Najim and A. S. Poznyak. *Learning automata: theory and applications*. Elsevier, 2014.

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

[18] M. S. Obaidat, S. Misra, and G. I. Papadimitriou. Guest editorial: Adaptive and learning systems. *Trans. Sys. Man Cyber. Part B*, 40(1):2–5, February 2010.

[19] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis. Guest editorial learning automata: theory, paradigms, and applications. *IEEE Transactions on Systems Man and Cybernetics Part B*, 32(6):706–709, 2002.

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

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

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

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

[24] A. S. Poznyak and K. Najim. *Learning automata and stochastic optimization*. Springer Science & Business Media, 1997.

[25] F. Seredyński. Distributed scheduling using simple learning machines. *European Journal of Operational Research*, 107(2):401–413, 1998.

[26] A. Shirvani. *Novel Solutions and Applications of the Object Partitioning Problem*. PhD thesis, Carleton University, Ottawa, Canada, 2017.

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

[28] M. A. L. Thathachar and PS Sastry. A class of rapidly converging algorithms for learning automata. In *IEEE Int. Conf. on Systems, Man and Cybernatics*. IEEE, 1984.

[29] M. A. L. Thathachar and P.S. Sastry. Networks of learning automata: Techniques for online stochastic optimization, secaucus, 2003.

[30] M. Tsetlin. On behaviour of finite automata in random medium. *Avtomat. i Telemekh*, 22(10):1345–1354, 1961.

[31] M. TSetlin et al. *Automaton theory and modeling of biological systems*. Academic Press, 1973.

[32] J. D. Ulman. *Principles of Database Systems*. Computer Science Press, 1982.

[33] C. Unsal, P. Kachroo, and J. S. Bay. Simulation study of multiple intelligent vehicle control using stochastic learning automata. *Transactions of the Society for Computer Simulation*, 14(4):193–210, 1997.

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

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

[36] 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.

[37] 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.