Novel Solutions and Applications of the Object Partitioning Problem

By
Abdolreza Shirvani

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy

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

May 2018

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

Novel Solutions and Applications of the

Object Partitioning Problem

submitted by

Abdolreza Shirvani

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

Dr. B. John Oommen
(Thesis Supervisor)

(External Examiner)

Carleton University
May 2018

ABSTRACT

This thesis considers the fundamental problem of "partitioning" which is all-pervasive in computer science. It has applications in "Big Data" because the vast amounts of data encountered in "Big Data" applications cannot be processed in a single block, but are better analyzed when it is partitioned in various monolithic units. It also has direct applications in numerous areas including databases, process scheduling, mapping and image retrieval. In this research we consider a specific instantiation of the Object Partitioning Problem, namely the Equi-Partitioning Problem (EPP), in which the partitions are equi-sized. In particular we concentrate on the various Learning Automata (LA)-based solutions. In this regard, the Object Migration Automata (OMA), and its variants have been the benchmark solutions.

The thesis thoroughly investigates the structural aspects of the OMA such as its internal grouping-based transitions, structure, initialization methods, convergence criteria and its recorded deficiencies. Rather than solely consider how the *transitions* can be designed, we specifically investigate how the well-known Pursuit paradigm, that has been recently invoked to enhance the field of LA, can be used to optimize the OMA. To do this, we propose two models for the Environment when it is noisy and noiseless, and show that the LA's convergence can be improved if we can discern the so-called "*divergent*" queries, namely, those which cause the LA to be sluggish. We have thus devised a technique that recognizes the "*divergent*" pairs and excludes them from the learning cycle, and to thus effectively pursue the optimal solution. This has resulted in the Pursuit OMA (POMA), whose performance is sometimes nearly two times faster than the original OMA.

Other researchers have observed and corrected the deadlock phenomenon in the original OMA, and have thus designed the so-called Enhanced OMA (EOMA). In this thesis, we have shown how the Pursuit phenomenon can be incorporated into the EOMA to yield the Pursuit EOMA (PEOMA), which, in some cases, is more than *forty* times faster that the traditional OMA.

Thereafter, we have observed that all the previous versions of the OMA were not able to include the property of transitivity. To include this phenomenon, we have also extracted the relations among the objects by utilizing the information in the

Pursuit matrix. The newly-proposed method that incorporates Transitivity, the so-called Transitive PEOMA (TPEOMA) outperforms the PEOMA by a factor of two, and is sometimes about *ninety* times faster than the OMA.

To finally demonstrate the power of the OMA-based paradigm in a real-life application, we have also incorporated it to resolve the outlier detection problem in Noisy Sensors Networks (NSNs).

ACKNOWLEDGEMENTS

I will, first and foremost, thank God, my Creator, for giving me the strength and patience to see this work through to its fruitful conclusion. This has been no small endeavor, which has encountered numerous obstacles, highs and lows. He has helped me all along.

I am also grateful to my father, Mohammad, and my mother, Forough, for being a constant stream of love and support during this period of my life. They have been a rock, without which I could not have made it this far.

I reserve special thanks to my thesis Supervisor, mentor, and friend, Prof. B. John Oommen. He has always allowed me to call him "John". John is like a father to me. I have learned a lot from him – not just academically, but also in character and virtue. He has left an indelible mark on my value system and personality. He is a role-model worthy of emulation.

Many thanks to my brother, Alireza, for being there for me. He is a bulwark and a friend whom I can call on at any time, and anywhere. Further, I want to thank Prof. Doron Nussbaum and Prof. Mark Lanthier for their significant assistance in ensuring that I received financial support throughout my program. Their help is a big part of my being able to cross the finish line. I am truly grateful.

I also want to thank my good friend, Vojslav (Voja), my labmate and roommate for a period of time. Voja was always willing to assist me at any time, and, indeed, is a true friend. I also thank Mohammad Ebrahimzadeh, my roommate, from whom I have learned a lot.

Finally, I want to thank the various friends that I made on this journey. They have made the entire experience worthwhile and enriching.

Contents

List of Figures

IMPORTANT ACRONYMS

AI: Artificial Intelligence
ASELA: Absorbing Stochastic Estimator Learning Algorithm
BAM: Basic Adaptive Method
EOMA: Enhanced Object Migration Automaton
EPP: Equi-Partitioning Problem
FSSA: Fixed Structure Stochastic Learning Automata
HS: Heuristic Search
LA: Learning Automata
MAP: Micro Aggregation Problem
ML: Machine Learning
MLE: Maximum Likelihood Estimator
NSN: Noisy Sensor Network
OMA: Object Migration Automaton
OPP: Object Partitioning Problem
PE: Partition Evaluator
PEOMA: Pursuit Enhanced Object Migration Automaton
POMA: Pursuit Object Migration Automaton
RE: Random Environment
RS: Reputation System
SDC: Statistical Disclosure Control
SMA: Stochastic Migration Algorithm
SELA: Stochastic Estimator Learning Algorithm
TPEOMA: Transitive Pursuit Enhanced Object Migration Automaton
VM: Virtual Machine
VQ: Vector Quantization
VSSA: Variable Structure Stochastic Learning Automata

"Not everything that can be counted counts, and not everything that counts can be counted."

Albert Einstein

1 Introduction

1.1 Preamble

We are living in an era in which data is being produced at an increasing, and almost unbelievable rate. Almost 90% of the data available on the planet has been created within the past few years [22]. 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"

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".

1.1.1 Important Aspects of Partitioning

A military strategy that was used for many millennia is one that is referred to as "divide and conquer". To win a war over a large country, military strategists have divided the country into smaller areas. Having conquered *these* areas, the country, in its entirety was defeated.

This same principle has been devised in designing efficient algorithms. Indeed, to sort a large list, one could recursively divide it into two sublists, sort them individually, and then merge the sorted lists in linear time. We can view this simple phenomenon as an example of what this thesis concentrates on, i.e., "partitioning".

Let us suppose that we are given a large amount of data, which requires numerous sectors of the associated memory. The question then is one of knowing which part of the data should reside on a particular sector, and how the data has to be partitioned so that it is optimally stored. The answer to this question requires one to determine what the "optimal" partitioning is, and this, of course, depends on the criterion for optimality. It is not unreasonable to assume that data elements that are fetched together should reside in the same sector. Alternatively, one could require that the data elements are placed together based on their time stamps or ownerships.

The issue that is of primary significance is that the data that is placed together must have some property of being similar or cohesive. Indeed, it is desirable to divide the data into smaller partitions so that the chunk of data falling within any partition is "related" to the rest of the data associated with it.

Partitioning need not necessarily have to do with data itself. Consider a database with many attributes. It could be advantageous if the attributes (and not the data itself) that are similar, are collected within the same sub-relation characterized by, for example, a primary and a secondary key. Similarly, it could be advantageous if the books in a library are placed together so that the books in "Mathematics" are all on the same floor, and hopefully, close to the books in "Engineering".

When we consider partitioning as a general tool, we see that it has all-pervasive-implications and applications. This thesis considers how "objects" (i.e., data elements, attributes, books etc.) can be partitioned, and the results that we obtain are, in our opinion, significant. We preface this by observing the fact that the partitioning problem is, in and of itself, *NP*-hard. In spite of this, the adaptive algorithms that we have devised have succeeded in solving it with astonishingly few operations.

1.1.2 Overview: The Partitioning Problem

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. As mentioned above, it is *NP*-hard, and so no general polynomial-time solution can 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 briefly survey this field in a subsequent chapter. We shall aim to resolve the partitioning problem using the tools available in LA.

1.1.3 Motivation of the Thesis

To achieve partitioning in unknown domains, researchers have incorporated models that use hypothetical objects to represent the true data elements, and to infer the partitioning based on the former. These hypothetical objects are manipulated by the respective algorithms, and by processing *these*, one minimizes the cost associated with migrating the *physical* objects themselves. This mode of operation is known in the literature as the Object Partitioning Problem (OPP). The application of this paradigm is significant, because it has been known to reduce the access time and the computations involved in processing the relevant data. The OPP has been studied since the 1970's. Because it is *NP*-hard, the original solutions were very time consuming. A survey of the methods proposed is presented in Chapter 3.

A near-optimal solution was presented *via* the so-called Object Migrating Automaton (OMA) paradigm. This involves the core of our research. As we shall show, it has been successfully applied to many real-world scenarios, and, indeed, its computational overhead remains efficiently within the boundary of present-day limitations. By way of example, the OMA can be used to assign *Processes* to *Processors* within a computer network. This problem focuses on optimally utilizing the processors, and it is achieved by collectively maximizing the load on all of the processors, and by thus executing the given task in the fastest possible manner. We mention that apart from the conceptual contributions made in this thesis, we also hope, in the future, to use our proposed solutions to the *Processes/Processors* problem alluded to here.

The OPP would be less difficult if the uncertain components associated with the problem were absent. In reality, its complexity stems from the uncertainty in the underlying phenomena. Indeed, the fact that we are dealing with random Environments renders the problem to be far more complex. The goal of any solution to the OPP, and in particular to the OMA paradigm, is that it should be able to determine the quality of the inferences of the objects that are associated with each other. In other words, it would be advantageous to infer the "*trustworthiness*" of the Environment that provides information about any specific object and about the objects that it "wants" to be associated with. This is also far from trivial, because as the designers of the algorithms, we cannot "control" the input that we receive.

First of all, the motivation for this thesis is to investigate the internal workings of the OMA from different angles. Even though it is more than hundred times faster than its predecessors, we would like to see how we can avoid deadlock situations and determine the condition when convergence has occurred.

One of the known deficiencies of the OMA is that when the problem is large, i.e., the number of objects and partitions are large, the probability of receiving a reward from the Environment is not significant. It would thus take the LA a *considerable number of iterations to recover* from an inferior or deadlock configuration. This property, that characterizes LA in general, is especially true for the OMA-based methods. In spite of the fact that various solutions have been proposed to remedy this issue for general families of LA, overcoming this hurdle is a completely unexplored area of research for conceptualizing how the OMA should interact with the Environment. Indeed, in all of the reported enhancements, all the OMA-like algorithms are still vulnerable to "bad input".

With some insight, one recognizes that none of the previous OMA-based solutions have taken any steps to distinguish these "bad inputs" and to resolve their occurrences. One of the primary motivations of this thesis is to see if it is possible to recognize such spurious input pairs as they appear in the input stream, and then to use this information to further catalyze the convergence.

As a research endeavor we are also motivated to see how the OMA, and for that matter, all partitioning algorithms can be enhanced by new phenomena that have not been used till now. There are, of course, many such phenomena, but the two most obvious ones are those of estimation and Pursuit, and transitivity. Inexpensive estimates have been used for at least two decades to enrich the field of LA. As far as we know, there is no evidence that they have been utilized in partitioning problems.

Additionally, there are three fundamental properties of relations that are central to characterizing them. These are the phenomena of reflectivity, symmetry and transitivity. It turns out that reflexivity and symmetry have been used previously in partitioning. One of the motivations of this thesis is to see if the phenomenon of transitivity can be included in a heuristic that can enhance the OMA.

Our research has been motivated by a desire to see how partitioning and OMA-based strategies can be applicable to new domains. We shall attempt to incorporate them into the domain of Noisy Sensor Networks (NSNs).

1.1.4 Problem Statement

The problem that we endeavor to resolve can be stated as follows: The OMA has been used as a benchmark algorithm to solve the Equi-Partitioning Problem (EPP). In the EPP, we are given W objects and R groups, and we would like to partition these objects equally among the groups so as to optimize some overall criterion. Our goal is to enhance the OMA using various concepts that have not been used in the literature till today, namely the joint statistics of the objects as recorded in the so-called Pursuit matrix, and the property of transitivity, that has not been used in the literature either.

The metric or measure of quantifying the quality of a solution, will be the number of query pairs that the system is provided with to attain to an accurate convergence, and also the the number of query pairs required for the machine to converge to its state(s) of final convergence, i.e., the the final partitioning. Since all the machines that are evaluated utilize the same criteria, the relative advantages and disadvantages of the various machines are accurately reflected using this metric.

1.2 Objectives of the Thesis

The objective of this research is to investigate how we can resolve the impediments of the OMA that were briefly mentioned in the previous section. Thus we shall aim:

- To enhance how the OMA operates internally;
- To reduce the inherent complexity of the solution proposed by the OMA by modifying its *structure*;
- To take advantage of the intrinsic properties of the Environment, and to use it to further enhance the OMA;

- To investigate possible application domains of the OMA and its enhanced derivatives.

To achieve these objectives, we have examined each of the above-mentioned items, and these have been fulfilled by considering the following components of the OMA:

- The initialization of the OMA;
- The generalization of the OMA's criterion for convergence;
- The resolution of the deficiencies encountered in exchanging abstract objects between the corresponding groups;
- The incorporation the inexpensive estimates and the Pursuit paradigm in all the prior versions of the OMA;
- Utilization of the phenomenon of transitivity to generate *inferred* queries while the Environment in and of itself, is dormant.

While much of what we have discussed above is related to the internal structure/configuration of the OMA, the partitioning problem itself, can appear at different levels of complexity as listed below:

- The number of objects to be partitioned and the number of partitions available;
- The Equi-Partitioning Problem (EPP), in which there are equal number of objects in each group;
- The Proportional Partitioning Problem (PPP), in which the number of objects in groups have common divisors;
- The Unbalanced Partitioning Problem (UPP), where there are no regularities between the number of objects in every group.

This thesis will only be concerned with the first two of these items while the last two items will be considered for future work.

In order to understand how we can take advantage of the properties of the Environment to improve the OMA, one of the objectives of the thesis is to invoke a completely different strategy than what has been currently used, and to thus mitigate the issues mentioned in the previous section. Since the Environment's behavior is unknown, the recognition of the relevance between the elements in the query pairs becomes crucial when it concerns minimizing the influence of the incorrect (referred to as the "divergent") pairs, and maximizing the properties of the correct inputs (i.e., the "convergent") pairs. In addition to identifying valid and invalid pairs, we provide a method to *infer* the existence of a similarity relation between the objects that have *not* been accessed yet by utilizing an *active* (as opposed to a passive, or in a dormant manner) strategy. To carry out all these, our objective is:

- To analyze the Environment's behavior in general;
- To additionally utilize an effective, yet efficient, mechanism to *estimate* how the Environment functions;
- To integrate this estimation technique into the operations of the OMA and all its derivatives;
- To extract tacit relations between objects by utilizing the obtained estimates.

Since the OMA has been deployed successfully in solving numerous real-life problems, clearly, any improvement that is useful to mitigate any of the above-mentioned handicaps, can be applied to enhance the solution to the corresponding application domains.

1.3 Contributions of the Thesis

In the context of what we have discussed in the previous two sections, we now state the contributions of this thesis. After a comprehensive survey of the field in Chapter 2, the contributions of Chapters 3 to 6, are the following:

- We have proposed a model for the Environment when it is noisy and noiseless, and studied the impact of the so-called "divergent" queries on the convergence rate of the OMA.
- We have devised a technique that has been incorporated into the OMA in order to recognize the divergent pairs and to exclude them from the learning cycle, and to thus effectively pursue the optimal solution.
- We have investigated the performance of the OMA equipped with the latter strategy, and designed the Pursuit OMA (POMA). The POMA is, in some cases, nearly *ten* times faster than the traditional OMA. The novel results reported in Chapter 3 have been published in [106], and its extended journal version in [103].
- We have applied the same strategy to the Enhanced version of the OMA (EOMA) to yield the PEOMA. The PEOMA is, in some cases, more than *forty* times faster that the traditional OMA. The novel results reported in Chapter 4 have been published in [104], and its extended journal version has been revised after a peer review and is currently being considered for publication [105].
- We have also taken a bold step and studied the transitivity properties of the Pursuit paradigm. The proposed method, the Transitive PEOMA (TPEOMA) outperforms the PEOMA by a factor of two, and is about *ninety* times faster than the original OMA. The results reported in Chapter 5 have been accepted for publication [107].
- To demonstrate the performance of our newly-proposed methods, we have deployed them in the study of NSNs. We have clearly documented the advantages of using our OMA-based schemes.

1.4 Hardware/Software Used for the Simulations

Before we move into the body of the thesis, in the interest of completeness, we state the details of the platforms on which the simulations in this thesis were done:

- The hardware used for the simulations in the thesis was an Intel Core i5-7200U CPU with 8GBs of physical memory installed;
- The software used for the simulations in the thesis was 'R' with a version 3.X (3.4.4 and before). This also supported the graphics packages.

1.5 Organization of the Thesis

This thesis is organized as below:

Chapter 2 is an overview of the background material that this thesis works with, and it particularly emphasizes the field of LA. The chapter covers the essential mathematical theory that fathoms the concepts that we will use. In particular, we will emphasize the families of *Estimator* algorithms and the *Pursuit* paradigm.

Chapter 3 overviews the prior solutions to the OPP and specifically focuses on the OMA-based methods. It provides various examples of the OPP where the OMA has been employed. This chapter also discusses the unvisited aspects of the Environment, and how they can impact the convergence of the OMA. Finally, the primary contribution of this chapter involves incorporating a new concept into the original OMA algorithm, i.e., utilizing the *Pursuit* paradigm to yield the POMA. The chapter also includes the relevant experimental results demonstrating the power of the POMA.

Chapter 4 presents the EOMA, i.e., the Enhanced version of the OMA, and formalizes the deficiencies of the original OMA that have been mitigated. The chapter then shows how we can combine the *Pursuit* concept with the EOMA to yield the Pursuit Enhanced OMA (PEOMA).

Chapter 5 first examines how we can improve the Pursuit principle. We then provide the theory and the necessary background to introduce the phenomenon of transitivity into OMA-based schemes, and thus pioneer the concept of *Inferred* queries by proposing the Transitive PEOMA (TPEOMA). By testing the TPEOMA in the benchmark environments, we document how it outperforms the previously presented methods.

Chapter 6 includes an application within the field of NSNs. After reviewing the necessary background, we incorporate the methods proposed in this research to resolve the NSN problem.

Chapter 7 concludes the thesis. It summarizes the results of each chapter and records the problems that we intend to study in the future, namely, the Unbalanced Partitioning Problem (UPP) and Proportional Partitioning Problem (PPP).

"What has been will be again,
what has been done will be done again;
there is nothing new under the sun."

King Solomon

2

Learning and Learning Automata

2.1 Introduction

2.1.1 Learning and Perspectives on Learning

Learning, from a psychological perspective, can be described as the process of maximizing the coherence of an individual's response to the stimulus that he receives from the surrounding domain that he interacts with. It could be readily observable, as in the case when one learns how to swim. Alternatively, the learning can be seen to lie "below the surface" such as in improving one's swimming skills. Maximizing the reward from the domain is the task we aim to achieve. For example, in the case of swimming, the ultimate goal may be to attain societal recognition or to gain a sense of self satisfaction and accomplishment. From our viewpoint, learning is not merely an issue of acquiring knowledge and skills, but also one of gleaning values,

attitudes and possibly, emotional reactions. The authors of [91] define learning as a long-term change in mental portrayal or associations as a result of one's experience. They classified it into three categories:

1. Learning as a long-term change. Here, the phenomenon is not necessarily temporal nor episodic.

2. Mental portrayal or associations. This implies that the brain is involved in the process.

3. Change as a result of an experience. From this perspective, learning is the outcome of an experience and not the result of external or analogous psychological habits.

Although there is a debate as to what would be the best definition for learning and when learning is fully accomplished, scientifically, we would like it to be qualified and quantified through an objective and systematic process. Learning can be described by the effective *principles* that are involved, and it is desirable that the factors that have the most influence on the learning process are clearly evident. After learning has been achieved, a behavior with an attractive or a motivational property (a reward) will be more prevalent compared to a non-rewarding behavior. Psychologists refer to this as the *reinforcement principle* [91].

The computational theory of learning emerged out of the psychological experiments conducted in the associated studies, which eventually resulted in extensive and diverse hypotheses about the phenomenon of learning and behavior [115]. In contrast to the field of experimental psychology, the goal of the computational approach is to develop principles and formulate them for the synthesis of learning systems. In the computational paradigm associated with learning and machine intelligence, a machine should be capable of drawing human-like conclusions within a confined or a specific domain. It is also sometimes expected that the machine behaves adaptively in response to the changes made in the Environment[1] to make better decisions. Thus,

[1]Throughout this thesis, we shall treat the Environment as an entity in and of itself, and we thus choose to "capitalize" it.

in modeling a learning principle, we simplify the domain to the point where the Environment of the machine is similar to those in the experiment, and by expressing the influential factors and the target goals as mathematical expressions, we prescriptively quantify and quality the learning algorithms. Evaluating the effectiveness of the software agent and the learning method used in this ideal situation can be construed as building a semantic relation between the cause and the effect following an adaptive behavior based on it.

Building a computational model for learning can be studied statistically, or it can be investigated using Machine Learning (ML) techniques. From the statistical perspective, one investigates the validity of the mathematical model whose parameters are to be estimated, while the ML approaches focus on the validity of the estimated parameters. In ML, we do not necessarily have an *a priori* justification, and so the model may differ in various ways such as in the:

- Sets of parameters or features;
- Complexity of the model (population size in evolutionary methods, number of actions in Learning Automata, number of hidden Layers in a neural network, etc.);
- Initial values for the associated variables and/or for the stopping criteria.

Selecting a statistical model from among many candidates involves a form of estimation of a physical quantity, and is referred to as *model selection*. Typically, the parameters of a specific model do not represent any useful intuition, and the concern is not one of estimating the parameters accurately, but rather of being able to increase the generalization capabilities of the model, which is the accuracy of the machine's capabilities on previously-unseen data.

2.2 The Learning Automata (LA)

From the behavioral perspective of psychology, the subject is often presented as a stimulus, and the response to the stimulus produces the "behavior". Thus, without

a stimulus one cannot assume the existence 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)[2]

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".* [67]

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 [9] and Bush [19]. In the early 1960's and 1970's, Russian scientists such as Tsetlin, Krinsky and Krylov [117], and Varshavskii [122]

[2]It is pertinent to differentiate between the fields of LA and Reinforcement Learning(RL). The field of LA was a precursor to the field of RL and it can be accurately considered as the skeletal form of RL. RL methods are divided into *value-based iteration* methods and *policy-based iteration* methods. The value-based RL schemes incorporate a state-action value function to determine which action should be performed in a given state. When the outcome is determined, the value function is used to identify the policy which describes the behavior of the agent. Contrary to the value-based approaches, policy-based iteration schemes (such as LA) operate and learn directly in the policy domain.

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 [65], [75], [85] and [93]. LA have also been applied successfully to address enterprise control problems such as network call admission [10], traffic control and quality of service routing [11] and Intelligent Vehicle Control [121] and [123]. In the more general field of AI, LA have been applied to Neural Network Adaptation [64], training Hidden Markov Models [45] and have been used to address the Distributed Scheduling problem [101]. For a comprehensive review of the applications of LA we encourage the reader to [52], [66], [96], [115] and [116], and the more recent survey journal issues in the field, i.e., [73] and [74].

The material covered in this chapter includes only the background details required for the rest of the thesis. We do not intend to over-complicate issues by including irrelevant analytical and mathematical results. Also, in our discussions, we will not include the fine details and extensions of LA that involve an infinite number of actions, or the various stochastic optimizations methods that LA can be applied to. The goal of this chapter is to provide a brief, yet solid, foundation, and a sufficient overview of the field, so that the reader can understand the concepts introduced in the subsequent chapters.

2.2.1 Formalizing Learning Automata

Definition 2.1. Narendra and Thathachar [68] have defined an LA as a quintuple $\langle A, B, \Phi, F(\cdot,\cdot), G(\cdot) \rangle$, where:

(i) $\Phi = \{\phi_1, \phi_2, ..., \phi_s\}$ represents the **set of states**, and $\phi(t)$ refers to the automaton's state at the time instance t.

(ii) $A = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ represents the set of possible **actions** (responses), and $\alpha(t)$ represents the action chosen by the automaton at the time instant t .

(iii) $B = \{ \beta_1, \beta_2, ..., \beta_m \}$ represents the set of **inputs** (stimuli), where $\beta(t)$ is the input at the time instant t. If we associate $B = \{0, 1\}$, we interpret $\beta = 0$ to the scenario when the Environment *rewards* the automaton. Similarly $\beta = 1$ represents the case in which the automaton has been *penalized* by the Environment. Throughout this thesis, we will only deal with the scenario where $B = \{0, 1\}$.

(iv) $F(\cdot, \cdot) : \Phi \times B \to \Phi$ represents the LA's **transition function**. It defines a set of rules that associates the present state of the LA, $\phi(t)$, and the input $\beta(t)$ to yield the next state $\phi(t+1)$. F is either a deterministic/stochastic function such that $\phi(t+1) = F(\phi(t), \beta(t))$.

(v) $G : \Phi \to A$ is the **output function** yielding the LA's output, $\alpha(t)$, as a deterministic or a stochastic function of the current state, $\phi(t)$, where, $\alpha(t) = G(\phi(t))$.

When all the sets Φ, B and A are finite, the LA is said to be a finite automaton.

The LA interacts with a stochastic Random Environment (RE), $\mathbb{E}$, whose response is based on the action chosen by the LA. $\mathbb{E}$ is mathematically expressed as a triple $\langle A, C, B \rangle$ described as below:

(i) $A = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the set of actions.

(ii) $B = \{\beta_1, \beta_2, ..., \beta_m\}$ is the Environment's output. If $m = 2$, $\beta = 0$ is referred to as a *reward*, and $\beta = 1$ is the *penalty*.

(iii) $C = \{c_1, c_2, ..., c_r\}$ determines the penalty probabilities of the Environment, where $c_i = \Pr[\beta(t) = 0 | \alpha(t) = \alpha_i]$. The quantity, c_i, is associated with the Environment's action, α_i and is referred to as the action's penalty probability.

The stochastic learning process is an iterative learning scheme which involves $\mathbb{E}$, the Random Environment, and the LA. The interaction between the RE and the LA is depicted in Figure 2.1. The learning consists of two steps:

- **Policy Evaluation**: The LA evaluates $\mathbb{E}$'s feedback and interprets it as a reward or a penalty.
- **Policy Improvement**: Based on the feedback received, the LA will adaptively choose the subsequent action so as to maximize the probability of receiving a reward.

The LA chooses an action $\alpha(t)$ at time t from the set of given actions. The selected action will then be stochastically evaluated by $\mathbb{E}$, and in response to $\alpha(t)$, it will provide the LA a reward (denoted by 0) or penalty (denoted by 1). The LA algorithm is designed in a such way that it minimizes the penalty by iterating the above-mentioned policy. Given enough number of iterations, the LA will hopefully converge to the configuration that yields the maximum reward.

To formalize this, we define the vector $\boldsymbol{P}(t) = [\, p_1(t), p_2(t), \cdots, p_r(t) \,]^T$, where $p_i\,(t)$ represents the probability of the LA choosing a specific action α_i at time t. Thus:

$$\begin{aligned} &p_i(t) = \Pr[\, \alpha(t) = \alpha_i \,] : \; i = 1, \cdots, r, \text{ and} \\ &\sum_{i=1}^{r} p_i(t) = 1 \; \forall t. \end{aligned} \tag{2.1}$$

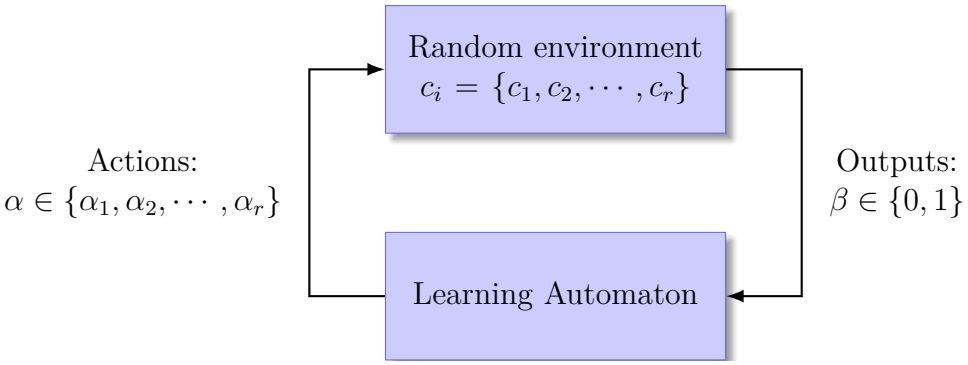


Figure 2.1: The Automaton-Environment Feedback Loop.

The *average penalty*, $M\,(t)$, can be calculated at time t given its action probability

vector $\boldsymbol{P}(t)$ as:

$$\begin{aligned} M(t) &= \boldsymbol{E}\left[\,\beta(t)|\boldsymbol{P}(t)\,\right] = \Pr\left[\,\beta(t)=1|\boldsymbol{P}(t)\,\right] \\ &= \sum_{i=1}^{r} c_i p_i(t). \end{aligned} \tag{2.2}$$

Accordingly, the average penalty for the *pure-chance* LA is:

$$M_0 = \frac{1}{r}\sum_{i=1}^{r} c_i. \tag{2.3}$$

Definition 2.2. An LA which asymptotically has a better performance than the pure-chance LA is said to be an *Expedient* LA. In such a case,

$$\lim_{t\to\infty} \boldsymbol{E}\left[M(t)\right] < M_0. \tag{2.4}$$

Definition 2.3. Absolute expediency is the corresponding monotonic property of the LA, where the Expediency is monotonic in the expected sense. In such case:

$$\boldsymbol{E}\left[\,M(t+1)|\boldsymbol{P}(t)\,\right] < M(t). \tag{2.5}$$

Definition 2.4. If an LA satisfies the following condition, it is said to be *Optimal*:

$$\lim_{t\to\infty} \boldsymbol{E}[\,M(t)\,] = c_l. \tag{2.6}$$

where $c_l = \min_i\{\,c_i\,\}$.

Definition 2.5. Since Optimality is unattainable with finite memory, we say that an LA is ϵ-optimal if:

$$\lim_{t\to\infty} \boldsymbol{E}\left[\,M(t)\,\right] \leq c_l + \epsilon, \tag{2.7}$$

where $\epsilon > 0$ is an arbitrarily small value. Of course, an LA can only be ϵ-optimal asymptotically.

2.3 Structures of the LA

The structure of the LA is defined by the types of its transition function, $F(\cdot,\cdot)$, and its output function, $G(\cdot)$, introduced earlier. A *deterministic* LA is one in which both

the transition and output functions are deterministic. In such cases, we can uniquely resolve its next state and action from the LA's current state. A *stochastic* LA has either or both the stochastic transition and output functions stochastic. Thus, given the transition function $\boldsymbol{F}(\cdot,\cdot)$ we can only determine the probability of each state being the destination state from the present state. If the conditional transition probability matrices have the form $\boldsymbol{F}^{\beta_1}, \boldsymbol{F}^{\beta_2}, \cdots, \boldsymbol{F}^{\beta_m}$, then each conditional probability matrix $\boldsymbol{F}^{\beta} : \beta \in \boldsymbol{B}$ is a $s \times s$-dimensional matrix with each of its elements, f_{ij}^{β}, having the following form:

$$f_{ij}^{\beta} = \Pr\left[\phi(t+1) = \phi_j \,|\, \phi(t) = \phi_i, \beta(t) = \beta\right] : i, j = 1, 2, \cdots, s. \tag{2.8}$$

Each entry f_{ij}^{β} of the matrix $\boldsymbol{F}^{\beta}$ in Equation (2.8) is the transition probability of the LA moving from state ϕ_i to state ϕ_j.

2.3.1 The Classes of LA

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 [118], and to the book by Narendra and Thathachar [68], as it has also explored these machines in fair detail. The authors of [68] 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 [35], [112] discussed later. Since we will

extensively work with all of them, we will now provide a brief overview of all the above-mentioned families.

2.3.2 Fixed Structure Stochastic Learning Automata (FSSA)

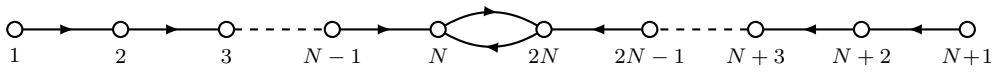
Every FSSA can be described using a Markov chain and the corresponding state transition probabilities could be specified in terms of the *reward/penalty* probabilities of the Environment [66]. Tsetlin's pioneering LA, described in [117], is a typical linear LA with $2N$ states, and it has been proven to be expedient. Further, when the probability of receiving a penalty for the best action is less than $\frac{1}{2}$, it is ϵ-optimal. Tsetlin also proved the expediency and ϵ-optimality for general cases which included multiple actions, and some non-stationary environments.

Since all the FSSAs reported in the literature use analogous approaches to track the past behavior, we briefly summarize some of these for the 2-action scenarios, where all of them possess $2N$ states, $\{\phi_1, \phi_2, \cdots, \phi_{2N}\}$, and two actions, α_1 and α_2. The states $\{\phi_1, \phi_2, \cdots, \phi_N\}$ belong to action α_1, while the states $\{\phi_{N+1}, \phi_{N+2}, \cdots, \phi_{2N}\}$ correspond to action α_2.

We now describe the Tsetlin Automaton referred to as $L_{2N,2}$. For a favorable response from the Environment, the automaton's state goes one step deeper into the memory. On the other hand, on receiving a penalty, the automaton migrates towards the opposing action one step at a time. Figure 2.2 depicts the general state transition graphs for the $L_{2N,2}$ Tsetlin automaton.

The corresponding transition matrices $\boldsymbol{F}(0)$ and $\boldsymbol{F}(1)$ are given in Equation (2.9)

The state transition of Tsetlin's $L_{2N,2}$ on receiving a *reward*, $\beta = 0$



The state transition of Tsetlin's $L_{2N,2}$ on receiving a *penalty*, $\beta = 1$

Figure 2.2: The state transition graphs for Tsetlin's $L_{2N,2}$.

and Equation (2.10) respectively.

$$\underset{(2N\times 2N)}{\boldsymbol{F}(0)} = \left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 0 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 0 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & 1 & 0 \end{array}\right]. \tag{2.9}$$

$$\underset{(2N\times 2N)}{\boldsymbol{F}(1)} = \left[\begin{array}{ccccc|ccccc} 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & 1 \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 & 0 & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{2.10}$$

Due to the "block" forms of $\boldsymbol{F}(0)$ and $\boldsymbol{F}(1)$, Tsetlin proved that the $L_{2N,2}$ is ϵ-optimal whenever the probability of receiving a penalty for the best action is less than $\frac{1}{2}$ [67].

Following Tsetlin's work, different variants of Tsetlin's $2N$-state LA were proposed. Krinsky [49] presented a version in which the objects were migrated to the extreme end state on every reward, while the transitions on being penalized were the same as in the $L_{2N,2}$. Krinsky's machine was shown to be ϵ-optimal in all REs.

Krylov's approach was the following: On receiving a reward it changed the state one step at a time as in the $L_{2N,2}$. However on receiving a penalty, it moved, with probability $\frac{1}{2}$, as if the input was a $L_{2N,2}$ reward, and with probability $\frac{1}{2}$ as if it was a $L_{2N,2}$ penalty [50]. The Krylov automaton was also proven to be ϵ-optimal in all REs.

Expedient LA with symmetric state transitions were proposed by Kandelaki and Tsertsvadze. The reward transition matrix for these LA were block-diagonal [46]. By consolidating several states of the automaton into a so-called STAR structure, the authors of [103] devised an LA with arbitrary branches, which, in turn, could be interpreted as a variant of Tsetlin's generalized form [109], but rather with *probabilistic* state transitions.

The authors of [49] enhanced the paradigm proposed by Krinsky, and devised

an automaton which behaved the same in the case of a penalty, but on receiving a reward, the machine moved to the middle state rather than to the "extreme" end state [42]. This LA demonstrated a faster recovery in the case of random rewards, and was also proven to be ϵ-optimal. Jamalian *et al.* also proposed a method which behaved similar to the Krinsky LA on receiving rewards, but translated to the previous state with probability θ and the next state with the complementary probability, $1 - \theta$.

Aso *et al.* further proved in [7] that in stationary REs the automaton would select the action associated with the highest reward probability. Further, if the penalty probability was equal for a subset of the actions, the LA should be able to select them with equally-likely probabilities. In other words, they suggested that the state transitions would not be dependent on the actions taken, and that the number of actions had to be smaller than or equal to the number of states. They opted to compose two automata, one of which was responsible for deciding *when* to change the action, and the other responded to the output of the first automaton by selecting the new action to be chosen. This LA was called the *Cascade-Composition* automaton.

An automaton which simultaneously took advantage of the feedback received from several teachers (or REs) was first studied by Koditschek and Narendra in [48]. This included a two-teacher set-up which was introduced to provide the feedback for two actions, and where the REs agreed on the *ordering* of the actions but not on the penalty probabilities. These authors showed that such a set-up could take advantage of a more expedient strategy when any single action was concurrently agreed on by both the REs. The authors also concluded that increasing the number of REs would improve the performance, by increasing the confidence in choosing a superior action at the cost of a slower convergence rate.

The closely related two-armed bandit problem was studied by the authors of [24], where each action was associated with a single "arm". They obtained an upper bound for the expected penalty, which led to an ϵ-optimal automaton. Although their experiments needed a sequence of samples whose size was related to the number of states in the automaton, it was difficult to expand their design for scenarios with more than two actions. The ϵ-optimality of this scheme was further proven by Chandrasekaran and Lakshmanan who studied the more general cases in [51]. These authors were able

to obtain a loose upper bound for the error probabilities. Later, Yakowitz [129] constructed an absorbing FSSA which could process an arbitrary number of hypotheses, and which could also address sequential decision procedures.

As we will discuss in detail in Chapter 3, FSSAs are capable of solving extremely complicated combinatorial optimization problems, as proposed by Oommen and Ma in [83]. Their solution, referred to as the Object Migration Automaton (OMA), is the foundation for other general partitioning algorithms such as the k-partitioning algorithm by Fayyoumi and Oommen [26], and the algorithms that we will discuss in Chapters 3 and 4. The Equi-Partitioning Problem (EPP) has been further expanded by the authors of [84] by exchanging the two objects residing in the most external states, probabilistically.

The OMA has been also deployed to uniformly partition a graph into sets of equal sizes. The goal of the uniform graph partitioning problem is to minimize the overall cost of the edges connecting vertices in the various partitions.

We conclude this section by mentioning that the OMA is the basis of the schemes studied in a significant problem of this thesis.

2.3.3 Variable Structure Stochastic Learning (VSSA)

In contrast to the family of FSSA, the transition probabilities of VSSA are not constant through time and the action probabilities are updated at every time instant. Essentially, the transition probabilities, f_{ij}^{β}, and the output function g_{ij} are changed at every time instant, and so the action probabilities are updated continuously. The category of automata that are described in this section are the linear VSSA schemes, but one can easily extend the concepts to derive nonlinear updating schemes as well.

VSSA were first proposed by Varshavskii and Vorontsova [122]. They utilized Markov chains to adjust the transition probabilities. They receive a binary feedback from the RE, and if it is a reward, the action probability of the relevant action is increased. However, in the case of a penalty, this probability could possibly be decreased. Such an approach, in which the changes are linear functions of the action probability vector, is referred to as the *Linear Reward-Penalty*, (L_{RP}) scheme.

To formalize the L_{RP}, we shall first describe the general non-linear VSSA, and then see how the equations are obtained for the linear scenario. A VSSA with r actions, operating in a static RE with $\beta = \{0, 1\}$, can be specified as below, where n indicates the iteration or the time index, and $P(n)$ is the action probability vector at time instance n. Using this notation, if $\alpha(n) = \alpha_i, (i = 1, 2, \cdots, r)$, the updating for $\beta = 0$ and $\beta = 1$ are as in Equation (2.11) and Equation (2.12) respectively.

$$\begin{array}{ll} p_j(n+1) = p_j(n) - g_j[P(n)] & \text{when } \beta(n) = 0, \\ p_j(n+1) = p_j(n) + h_j[P(n)] & \text{when } \beta(n) = 1, \forall j \neq i. \end{array} \tag{2.11}$$

Since $\sum_{j=1}^{r} p_j(n) = 1$, this implies that:

$$\begin{array}{ll} p_i(n+1) = p_i(n) + \sum_{j=1, j\neq i}^{r} g_j[P(n)] & \text{when } \beta(n) = 0, \\ p_i(n+1) = p_i(n) - \sum_{j=1, j\neq i}^{r} h_j[P(n)] & \text{when } \beta(n) = 1. \end{array} \tag{2.12}$$

The functions g_j and h_j for $j = 1, 2, 3, \cdots, r$ in Equation (2.11) and Equation (2.12) are valid under the following conditions:

$$\begin{array}{ll} \textit{Assumption } 1: & \forall j, g_j \text{ and } h_j \text{ are continuous functions,} \\ \textit{Assumption } 2: & \forall j, g_j \text{ and } h_j \text{ are non-negative functions,} \\ \textit{Assumption } 3: & 0 < \sum_{j=1, j\neq i}^{r} \left[p_j(n) + h_j[P(n)] \right] < 1. \end{array} \tag{2.13}$$

In the specific case when, if $\forall j$, g_j and h_j are linear functions, we can use

$$\begin{array}{ll} g_j[P(n)] = a p_j(n) & 0 < a < 1, \\ h_j[P(n)] = \frac{b}{r-1} - b p_j(n) & 0 < b < 1. \end{array} \tag{2.14}$$

Thus, if $\alpha(n) = \alpha_i$,

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

$$\left.\begin{array}{l} p_i(n+1) = (1-b) p_i(n) \\ p_j(n+1) = \frac{b}{r-1} + (1-b) p_j(n), j \neq i \end{array}\right\} \quad \beta(n) = 1. \tag{2.15}$$

The L_{RP} scheme with $a = b$ is at best expedient. However if $b \ll a$, it becomes ϵ-optimal in the limit by enforcing the condition that a approaches zero.

This L_{RP} was also first implemented to deal with continuous data by McMurtry and Fu in [63]. They designed the updating of the action probabilities by utilizing the stationary action probability vector, and by updating it directly by observing the change in the transition probabilities of its intrinsic MC.

Another variant proposed by Shapiro and Narendra [102] is the *Linear Reward-Inaction*, (L_{RI}) scheme. The L_{RP} scheme given by Equation (2.15) reduces to the L_{RI} when $b = 0$. Viswanathan and Narendra, in [126], showed that this LA is ϵ-optimal in all REs. They also experimentally compared and analyzed the performance of two linear methods, the L_{RP} and the L_{RI} with the Varshavskii-Vorontsova's nonlinear quadratic scheme [122] and Vorontsova's non-linear two-action method [128]. In stationary REs, the optimal solutions outperformed the expedient schemes both in games and in hierarchical REs.

The P-model and S-model feedbacks were proposed by the authors of [21], where the former operates in a stochastic RE with a stochastic *binary* feedback, while in the latter, the random feedback strength is calculated within a unit interval. In their publication, they discovered and compared the convergence characteristics of the L_{RP} schemes as well as that of a few similar reinforcement methods. They concluded that the quadratic non-linear scheme displayed the best performance with respect to the algorithm's convergence. For the S-model case, they also showed that if one sacrificed the convergence speed, one could magnify the action probabilities by utilizing non-linear transformations on the strength of the feedback.

The optimal performance for the P-model was proposed by Vorontsova [128] who specified the conditions required for changing the transition probabilities for a two-action LA. Viswanathan and Narendra further studied the S-model in [127] by devising a unique method that extended the P-model to the S-model, and which simultaneously had an acceptable convergence accuracy and speed.

For stationary REs, Lakshmivarahan and Thathachar proposed two other generalized non-linear two-action methods which were both ϵ-optimal [54]. The method was capable of utilizing any non-linear probability update functions as long as they satisfied specific symmetric conditions. The ϵ-optimality of this method was proven

by the authors of [100] for any stationary RE. The multi-action extension was proposed by Shapiro and Narendra [102] by restricting the chance of receiving a penalty for the best action to be less than $\frac{1}{2}$, while for the other actions it is greater than $\frac{1}{2}$.

Norman studied how the learning behavior of the automaton evolves over time and showed that the gradual learning can affect the realizations of the probability paths which are normally distributed along the expected probability track [71], [72]. In his work, learning was considered to be gradual. His result was shown to be true if the action probabilities were updated in small step sizes, or if the learning rate was slow, or if the chance of updating was small, e.g., the chances of receiving a reward was small and the penalties were ignored. He examined "no penalty" policies for absorbing two-action linear schemes (such as the L_{RI}) and showed that if the learning steps are small, it reduces the chances of the LA converging to any of the wrong actions. This concept was generalized in [70] by eliminating the absorbing criteria for linear models and by additionally introducing a new concept in stochastic processes, i.e., that of *distance diminishing operators*, which was also the basis for proving the convergence of subsequent learning algorithms.

A generalized probability updating scheme were proposed by Lakshmivarahan and Thathachar in [55] for the P-model and in [56] for the S-model. The Absolutely Expedient schemes were proven to be ϵ-optimal in any stationary RE in [57]. They showed that if the updating rule is absorbing, it will eventually select only one of the available actions, which is equivalent to converging to one of the unit vectors. These authors also proved the trade-off between the convergence speed and the probability of converging to the best action. They achieved it by deriving a lower bound for the probability of converging to the best action.

Pozniak also worked on the concept of learning functionals in probability updating algorithms, and obtained sufficient conditions for convergence in [95]. The authors of [8] modeled a generalized updating scheme, and showed that the Absolute Expediency of the Q-model and the S-model could be seen to be direct consequences of the Absolute Expediency of the P-model.

Simha and Kurose proposed *two* algorithms for the S-model based on the phenomenon of relative strength [108]: The most-recently obtained feedback was stored

for the all actions, and the *first* of these algorithms would increase the probability for the action that had the largest stored feedback, and decrease the probability for all the other actions. Alternatively, the authors also proposed that the scheme could increase the probabilities of the actions *proportional* to the difference of their stored feedbacks to the mean feedback for all the actions, thus effectively leading to an increase in the selection probabilities for those actions with stored feedbacks above the average, and a decrease in the selection probabilities for the other actions.

2.3.4 Estimator and Pursuit Algorithms

As mentioned earlier, Absolutely Expedient algorithms are absorbing and there is always a small probability of them not converging to the best action. Thathachar and Sastry realized this phenomena and proposed to use Maximum Likelihood Estimators (MLEs). Such an MLE-based update method would utilize estimates of the reward probabilities in the update equations. They did this by maintaining a record of the number of times an action received a reward, and the number of times it was chosen. Clearly, the probability of getting a reward was then defined as the ratio between these quantities. Further, at every iteration, the estimated reward vector was also used to update the action probability vector, instead of updating it based only on the RE's feedback. In this way, the frequency of choosing the actions with higher reward estimates is increased and the chances of choosing actions with lower estimates are significantly reduced[3]. In doing this, Thathachar and Sastry proposed the family of estimator algorithms for the P-model in [115], and extended these concepts to the S-model in [113].

The *Pursuit* strategy of designing LA is a special derivative of the family of estimator algorithms. Pursuit algorithms "pursue" the currently-known best action, and thus to increase the action probability associated with the action that possessed the largest reward estimate. The pursuit concept was first introduced by Thathachar and Sastry in [114] and the corresponding LA was proven to be ϵ-optimal[4]. Its discretized

[3]Earlier, in [53], Lakshmivarahan and Thathachar had used probability estimation techniques to obtain a terminating criterion to stop the learning process. However, it was never used as a *part* of the learning process itself.

[4]All the proofs reported here for Pursuit and Estimator LA were flawed. This flaw was discovered

version was proposed by Oommen and Lanctot in [81], where they also proved the method to be ϵ-optimal. Oommen and Lanctot also discretized the original estimator algorithm proposed by Thathachar and Sastry in [113] in addition to proving the necessary conditions for it to be ϵ-optimal in [58]. The discretized version was shown by the same authors to outperform the continuous versions through simulation results. Agache and Oommen further completed their work by analyzing all the four discretized linear scheme combinations in [76], i.e., using the L_{RI} and L_{RP} paradigm combined with continuous and discretized probability update methodologies. Again the experimental results demonstrated the superiority of the discretized algorithms when compared to their continuous counterparts.

The vectorized updating rule for the estimator algorithm was proposed by Agache and Oommen in [3]. They utilized it to derive the formula for the generalized probability updating scheme obtained by applying unequal weights for the probabilities with greater reward estimate values. The discretized version of this scheme, the DGPA, was proposed in [2] by Agache and Oommen.

We conclude this section by mentioning that the enhancements that we provide for the OMA and its modified versions will specifically involve the pursuit concept.

2.3.5 Hierarchical LA

For problems that are large in size, i.e., the size of the number of actions to be considered by the LA is large, the probability of a specific action receiving a reward from the RE is not significant. This results in a low convergence rate. To address this issue, Poznyak, in [94], used a hierarchy of LA, which could only operate in a two-level system. Thathachar and Ramakrishnan constructed a hierarchical tree based system of LA, where only the actions located at the leaves interacted with the RE [111].

If all the automata at all levels of the tree were supposed to have the same number of r actions, the total number of actions supported by an L-level hierarchical system would be r^L, from which L can easily be determined. Thathachar and Ramakrishnan showed that the optimal number of actions per LA needed $r = 2$ or $r = 3$, with

in [61] and rectified in [135], [136], [137] and [138]. The reference [135] and [136] use the Baysian estimate of the reward probabilities instead of the MLEs.

the minimum number of updates occurring if the number of actions only had the digits two and three as factors. Thus, one would have to extend the original action set with some dummy actions so as to achieve this desirable property. Baba later extended this work in [12] to the multi-teacher situation where the automata at each level received multiple feedback signals from different REs for the suggested action.

Baba and Mogami [13] incorporated this principle into the paradigm proposed by Thathachar and Ramakrishnan [111] to obtain a hierarchical S-model automaton that was subsequently extended to the multi-teacher environment by the same authors in [14].

Ansari and Papavassilopoulos derived a scheme by applying Baba's GAE algorithm which chooses one action per RE [6] as opposed to the previous multi-teacher algorithms which required the automaton to choose one and only one action at every time instant. They showed that in this configuration, the automaton's probability vector is equivalent to the average of the probability vectors of k LA agents in the k analogous REs.

2.3.6 LA for Non-Stationary REs

Most of the automata algorithms have been developed for stationary REs, i.e., where the reward and penalty probabilities do not change over time. This is, unfortunately, not always a realistic model, and there have been some attempts[5] to formulate automata for non-stationary REs. One of the first of these was the two-level LA proposed by Narendra and Viswanathan for a periodically changing environment where the first-level automaton had "learned" the periods of the RE by tracking them as actions, thus triggering subsequent second-level automata tailored to the current period [69]. Baba and Sawaragi showed that the L_{RI} scheme is ϵ-optimal under a non-stationary RE where the penalty probability with the index of the action chosen by the automaton [15].

The weakness of the estimator algorithms in non-stationary REs is that the approximation may not realize the changes in the Environment, since the estimate of

[5]We should observe that all *ergodic* LA can be utilized for problems involving non-stationary REs.

the reward is only updated after the action is chosen. Vasilakos and Papadimitriou in [124], proposed a solution which utilized a pre-set constant time-frame of the most recent rewards received. This Ergodic Discretized Estimator Learning two-action automaton (EDEL) used a windowed reward estimate and discrete probability updates. However, in systems with many actions, or with large differences in the action probabilities, it can take many iterations between each time that an action is chosen, and thus even the most recent reward in the stored window is "old". Consequently, the RE may have changed its state since the time the newest reward was received.

The Stochastic Estimator Learning Algorithm (SELA), which was developed and analyzed by Papadimitriou in [125], aimed to overcome this by adding an "oldness vector" whose elements counted the number of iterations that had transpired since the last time the corresponding action was chosen. The "oldness" element for each action was defined by the variance of a zero-mean normal random variate that was added to this action's windowed reward estimate, before all the randomized estimates were used to decide which action was the best, and therefore subjected to an increase of its action probability. The intention was that actions that had not been tried for a while could still end up with the best randomized reward estimate. The SELA algorithm was subsequently proven to be ϵ-optimal for S-model random environments by Vasilakos and Papadimitriou [125].

Papadimitriou *et al.* also proposed an absorbing variant, ASELA, for stationary environments in [92]. Further, in such REs, there is no need to use a window of only the past few estimates. Thus, the standard deviation of the normal variates added to the MLEs was taken to be inversely proportional to the number of times an action was tried. In [30] the authors proposed a modified SELA algorithm in which uniformly-distributed random variates were added to the MLE. Its ϵ-optimality was also proven.

2.4 Conclusion

This chapter surveyed the field of computer science that is pursued in this thesis. We first investigated the concept of "Learning" from a psychological perspective, and

described how it has been borrowed into and applied in the area of Machine Learning (ML) algorithms. We then discussed a few aspects of ML and statistics such as *model selection* and *parameter estimation*.

After this, we proceeded to the main area of interest, namely, that of Learning Automata (LA), and demonstrated how strongly it is related to concepts in the theory of Learning. We showed how a general model of LA could be characterized and defined, both quantitatively and qualitatively, and how the learning characteristics of LA can be measured. We then proceeded to the two major classes of LA, namely the family of Fixed Structure Stochastic Automata (FSSA) and the family of Variable Structure Stochastic Automata (VSSA). Each of these families were considered in some detail. In these schemes, we also discussed the convergence and performance of the LA. The properties of these LA in multi-teacher and non-stationary random environments were also examined.

We then visited the topic of using Maximum Likelihood Estimators (MLEs) of the reward probabilities to enhance VSSA. This led to the families of Estimator and Pursuit LA.

In the following chapters, we will utilize these concepts to build novel approaches to address the OPP, which is one of the primary focuses of this thesis.

"When it is obvious that the goals cannot be reached,
don't adjust the goals, adjust the action steps."

Confucius

3

Object Partitioning Using LA-based Methods

3.1 Introduction

In this chapter[1] we will consider the Object Partitioning Problem (OPP). We will focus on a special case of the OPP, which was referred to as Equi-Partitioning Problem (EPP), where all the partitions have the same number of objects. We will summarize the previously-proposed significant solutions to the EPP. In Section 3.5 we will introduce a unique solution to the EPP, which is the Object Migration Automaton (OMA), and survey the applications in which it has been used. The OMA is the foundation for the best-reported solutions to the EPP.

[1]The novel results reported in this chapter, obtained from this research endeavor, have been published in [106], and its extended journal version in [103].

We then move on to the novel contributions of the chapter. We study and analyze the behavior of the OMA in two different Environments, i.e., for the ideal case when there is no noise, and for a non-ideal scenario where noise will distort the correctness of the data stream processed by the OMA, and how such a noisy system will affect its convergence. In Section 3.9, we argue that the data stream generated by the Environment can distract the OMA from converging. To rectify this, we propose incorporating the *pursuit* concept of LA, and we use this idea in Section 3.10 to introduce the *Pursuit OMA* (POMA). The experimental results of the POMA are given in Section 3.11. In Section 3.12.2 we present the results about how the POMA can improve the OMA algorithm in a variety of contexts, and show its superiority with regard to its performance and time-related behavior. Section 3.13 concludes the chapter.

3.1.1 Contributions of this Chapter

As we will argue in this chapter, the convergence rate of a partitioning algorithm is not as trivial as it initially appears. This is due to the existing interaction of the query generating system (the so-called Environment) and the various objects that it presents by means of queries. It is obvious that a high degree of uncertainty in the query pairs can drastically decrease the performance of the partitioning algorithm or even distract it from the desired result, which is, indeed, what occurs both in the initial and final stages of any partitioning algorithm. The main contribution of this chapter is that we demonstrate that the pursuit principle can be used as a remedy to mitigate the effects of such query pairs. It can enhance the algorithm with a negligible performance overhead.

As mentioned, the resulting machine is the so-called Pursuit OMA (POMA). We also demonstrate the power of the POMA in achieving superior results in less iterations, and with the same computational complexity as the OMA. Since the POMA utilizes a simple statistical policy matrix to accept/reject incoming queries, we argue that it operates in a "higher dimension" – utilizing various statistical measures to perceive the "trueness" or validity of the incoming pairs generated by the Environment.

It thus converges to a correct partitioning with sometimes almost half the iterations required for the OMA.

We are not aware of any other similar (or parallel) solution to the EPP.

3.2 On the Partitioning of Objects

We begin this section by introducing an abstract concept and proceed to relate it to real world physical entities. As mentioned earlier, partitioning real-life objects into certain subgroups, which, in turn, meet specific criteria, is a truly fundamental problem. For example, in certain applications, one desires 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 arena in which we operate.

Consider the problem of partitioning a set $\mathcal{A} = \{A_1, \cdots, A_W\}$ of W physical objects into $\mathcal{R}$ 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 [133]. 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.

3.3 The Object Partitioning Problem

If there exists a mutual relation between the real objects in the semantic domain $\mathcal{A}$ introduced in Section 3.2, 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 seek the plausible partitions only 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 [28].

For example, in a library Environment, $\mathcal{A}$ can be associated with the categories of books (Psychology, Economics, Computer Science, Mathematics, etc.). Realizing the partitioning of $\mathcal{A}$ will be equivalent to associating the physical locations of the books and documents in each of these individual disciplines [99]. Consider, for example, a student in Computer Science. The likelihood of him borrowing books in Mathematics and Computer Programming is higher than for a student who studies Philosophy to seek out books in Computer Programming. Thus it is advantageous for the library to have the books in Computer Science in the geographic proximity of the books in Mathematics. In this scenario, $\mathcal{A}$ would be the set of disciplines, while $\mathcal{O}$ will be the abstract objects, representing the disciplines, that are manipulated by the algorithm.

Another application can occur in database systems. Here $\mathcal{A}$ 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 [29] and [133]. 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.

All the problems described above were mentioned conceptually, and 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 was 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) described in the Section 3.4. 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. Since it is premature to go into this solution and these application domains in any depth, we shall describe the OMA in all its fine detail in Section 3.5, and the real-life applications in which the OMA has been used in Section 3.7.

In future, unless there is a need to do so, we will not distinguish between the physical and abstract objects.

3.4 The Equi-Partitioning Problem

The environmental factors which decree the optimality of a particular partitioning are dynamic and unpredictable in the long run. Therefore, a system that can quickly adapt the partitioning of objects, i.e., data, to the demands of a changing Environment would be very useful. We will consider a particular case of the OPP where the objects are grouped into equal partitions. This is referred to as the *Equi-Partitioning Problem* (EPP), where we are dealing with W objects and R classes, and for each class we have exactly $\frac{W}{R}$ objects. We attempt to solve the EPP by enforcing the following restrictions:

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

2. The queries appear with exactly two objects in each query[2].

The general case, where we have a variable number of objects in every class is far more complicated and will be discussed later.

In [83], the authors introduced the so-called *Object Migration Automata* (OMA) to address this problem. The OMA has many desirable characteristics such as quick convergence and simplicity. It can also be easily implemented. In the following subsection we will discuss the OMA, which is a fixed structure LA for the EPP.

3.4.1 Prior Solutions to the OPP/EPP

Several solutions have been proposed by researchers to solve the OPP. The most elementary solution to the OPP is to estimate the query statistics [23], 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 literature, e.g., [31] and [134]. Since the OPP is a known NP-hard problem [23], the optimal partitioning algorithm will have an exponential computational cost [134], 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 [31] and [32] 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.

The best known algorithm available for the OPP is presented in [83] and the second-runner is, probably, the Basic Adaptive Method (BAM) which was suggested by the authors of [134]. We will not review the advantages of this method over

[2]The question of extending this to multi-object queries will be reviewed later.

its predecessors here, as they are well-defended in [134]. The primary pre-cursor competitor to the BAM is a technique offered by the authors of [31] and [32].

In the following section, we provide a brief description of the technique initially proposed in [31] and [32] for the Attribute Partitioning Problem. Later, we will review the BAM algorithm in a scenario in which two objects are accessed per query. The case when more than two objects are accessed has also been handled in [134]. This assumption is necessary in presenting our model of computation, and in providing a proper foundation for the LA-based techniques discussed later[3].

In the intent of completeness, we will now briefly review each of these above-mentioned schemes.

3.4.2 A Hill Climbing Solution

A hill-climbing paradigm has been widely used in the literature for local optimization. In [31] and [32] Hammer *et al* have used these principles, associated with a step-wise approach, to address the problem of attribute partitioning [32], 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 Method:**
  The heuristic technique proposed in [31] and [32] is composed of two phases, namely, the Pairwise Grouping and the Attribute Regrouping. These two phases are applied alternatively until no further improvements can be obtained. These phases are applied equivalently and alternatively to the cost function in the same manner, until the cost of processing the transactions cannot be reduced

[3]When k objects are accessed in a query, we can consider it as a case of having $\binom{k}{2}$ *pairs* being accessed simultaneously.

by forming variations of the proposed partition.

Consider a scenario with a relation named Z [119] is to be partitioned and composed of w attributes. A partition of Z is defined as a set of blocks containing a subset of its attributes. Consequently, a valid partition of the attributes of Z is a candidate for the heuristic being used. The basic partition, where all w attributes are in separate blocks, will be chosen as the initial candidate for the heuristic.

The method starts by invoking the "Pairwise Grouping" phase. At every step in this phase, a set of variations is built by considering all possible pairwise groupings of the blocks of the current candidate. Thus, $\binom{w}{2}$ combinations can be constructed in the first step, since there are w attributes in Z. Thereafter, the PE is applied to all the obtained variations to achieve cost evaluations. The variation that produces the greatest improvement over the previous candidate replaces the latter. The algorithm continues until no further improvements can be achieved by applying the method.

The second phase, involving Attribute Regrouping, begins with the final candidate obtained from the first phase. In this phase, different variations of the current candidate are created at every step by taking an attribute from its current block to another block in the partition. Each variation consists of moving only a single attribute, and this thus results in the construction of $w \times (k-1)$ variations, where k is the block count of the present candidate solution. In an approach similar to the first phase, the costs of variations are evaluated by the PE, and the one yielding the greatest improvement over the previous candidate becomes the new current candidate. This phase also will continue until no further improvements can be achieved over the current candidate.

The method keeps applying the two phases until neither of them yields any improvement over the current candidate. At this point, the algorithm terminates, and the current candidate is selected as the best "sub-optimal partition".

- **Intuition of the Heuristic:**
  According to the authors of [23], the intuition behind this pairwise grouping can

be described as follows: We know that all the m attributes of R are initially located in different blocks. Further, the transactions that require only two attributes are handled with near-minimum cost, and those transactions that request k attributes, $2 < k \leq m$, are more costly to process. As the algorithm clusters attributes together, the processing cost of the two-attribute transactions becomes more expensive, since the unnecessary information obtained from other attributes co-existing in the same block, needs to be processed. Meanwhile the multi-attribute transactions become computationally less expensive as most (or all) of the requested attributes are co-existing in the same block. Eventually, the algorithm converges to an equilibrium point, where both of the above-mentioned costs will tend to moderate each other.

- **Experimental Results:**
  The authors of [23] claim that running a real-life simulation was not practical. They referred the reader to the papers [31] and [32] where some experimental results obtained by using the heuristics were provided. We too refer the reader to these references.

We proceed to discuss the Basic Adaptive Method (BAM) originally presented by Yu *et al* in [133] and [134], in the next section.

3.4.3 The Basic Adaptive Method (BAM)

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 [133] and [134], 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. In the intent of completeness, we briefly review the BAM. We refer the reader to [133] and [134] for further details, and for a more thorough analysis.

3.4.4 Abstract Objects

The BAM moves the abstract objects (alluded to earlier) rather than the real physical objects. Indeed, as explained in Section 3.2, the group of physical objects $\mathcal{A} = \{A_1, \cdots, A_W\}$, is associated with a set of abstract objects that are constantly being partitioned into classes. If the partitioned abstract objects differ from the physical partitioning, the set A will be re-partitioned to match the partitioning generated by the BAM, and this process is performed when the queries on the actual physical objects are not expected [23]. Every abstract object in the BAM contains only its identity and a position on the real line, as opposed to a real physical object which carries a lot of information. As this will also be the model we use, as mentioned in Section 3.2, the set of abstract objects will be specified by $\mathcal{O}$, and the set of the physical objects by $\mathcal{A}$. Thus, A_i represents the i-th physical object and O_i denotes its abstract counterpart[4].

3.4.5 The BAM Algorithm

The BAM assumes that a set of queries is specified against a set of physical objects $\mathcal{A} = \{A_1, \cdots, A_W\}$ in a manner that each query is composed of only two accessed

[4]Although the queries deal with physical objects, in the algorithms discussed in this thesis, we will only modify the abstract objects. Consequently, every time a set of physical objects is given, it automatically leads to the definition of a corresponding set of abstract objects and *vice versa*.

objects. Every abstract object O_i is represented by a real number X_i which represents its position on the real line for all $1 \leq i \leq W$. If two physical objects A_i and A_j are given in a query, we then move the corresponding abstract objects O_i and O_j towards their centroid by a small constant value Δ_1, where this centroid is computed by averaging the values of X_i and X_j as $\frac{(X_i+X_j)}{2}$. This tends to "push" the objects into the same group. Unfortunately, if this procedure happens too often, it will cause all the different classes to coalesce. To remedy this, we pick two random abstract objects O_p and O_q, where $1 \leq p, q \leq W$ and move them away from their centroid by a constant value, Δ_2. The authors of [134] describe the techniques[5] for specifying the values of Δ_1, Δ_2 and the X_i's. The algorithm will iterate these two steps for T iterations, and will return the suggested clusters based on the proximity of the X_i's. It does this by assigning the abstract objects that lie "close enough" to each other to the same cluster. The formal BAM algorithm is given in Algorithm 1.

3.5 The Object Migration Automata

The learning process of an LA works with the principles explained in Chapter 2. 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[6] to another. In the spirit of Figure 2.1, we can figuratively describe the schematic of the OMA by Figure 3.1.

[5] In the absence of any information, one can initialize the values of Δ_1 and Δ_2 to unity.

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

Algorithm 1 BAM algorithm

Input:

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

Output:

- A periodic clustering of the objects into R partitions.

begin
  Initialize X_i arbitrarily, where $-\infty < X_i < \infty$, for all $1 \leq i \leq W$.
  loop
    for a sequence of T queries do **do**
      Read query $\langle A_i, A_j \rangle$
      if $X_i < X_j$ **then**
        $X_i = X_i + \Delta_1$
        $X_j = X_j - \Delta_1$
      else
        $X_i = X_i - \Delta_1$
        $X_j = X_j + \Delta_1$
      end if
      Select randomly distinct indices p, q, where $1 \leq p, q \leq W$
      if $X_p < X_q$ **then**
        $X_p = X_p - \Delta_2$
        $X_q = X_q + \Delta_2$
      else
        $X_p = X_p + \Delta_2$
        $X_q = X_q - \Delta_2$
      end if
    end for
    Print out partitions based on proximity of $\{X_i\}$
  end loop
end

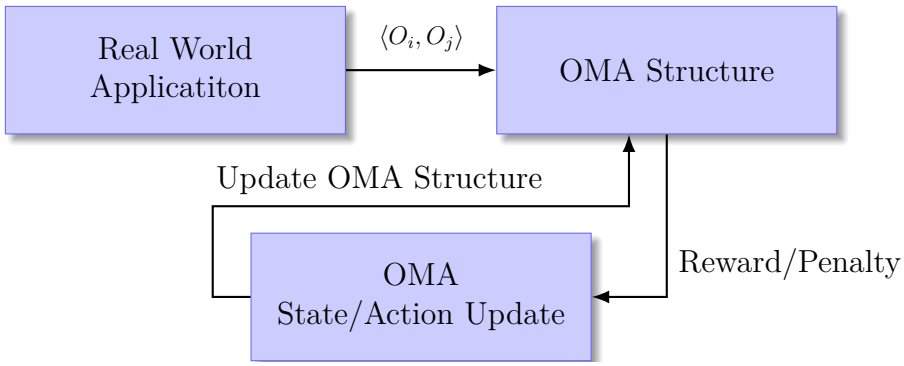


Figure 3.1: Schematic of the OMA in a real-world setting.

In the OMA, 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 Figure 3.2 **(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 Figure 3.2 **(b)**.

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 Figure 3.2 **(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 Figure 3.2 **(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 Figure 3.2 **(d)**.

The above rules are figuratively displayed in Figure 3.2, and Algorithm 2 presents the algorithmic representation of the technique described above. It invokes the procedures "ProcessReward" and "ProcessPenalty" given in Algorithms 3 and 4 respectively.

Algorithm 2 The OMA Algorithm

Input:

- The abstract objects $\{O_1, \cdots, O_W\}$.
- The number of states N per action.
- A stream of queries $\{\langle A_i, A_j\rangle, \text{ where } i \neq j\}$.

Output:

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

1: **begin**
2: Initialization of $\{\xi_p\}$ ▷ This is described in the text
3: **for** a sequence of T queries **do**
4: Read query $\langle A_i, A_j\rangle$
5: **if** ξ_i div $N = \xi_j$ div N **then** ▷ The partitioning is rewarded
6: Call ProcessReward($\{\xi_p\}, A_i, A_j$)
7: **else** ▷ The partitioning is penalized
8: Call ProcessPenalty($\{\xi_p\}, A_i, A_j$)
9: **end if**
10: **end for**
11: Print out the partitions based on the states $\{\xi_i\}$
12: **end**

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

Input:

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

Output:

- The next states of the O_i's.

1: **begin**
2: **if** $\xi_i \bmod N \neq 1$ **then** ▷ Move O_i towards the internal state
3: $\xi_i = \xi_i - 1$
4: **end if**
5: **if** $\xi_j \bmod N \neq 1$ **then** ▷ Move O_j towards the internal state
6: $\xi_j = \xi_j - 1$
7: **end if**
8: **end**

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

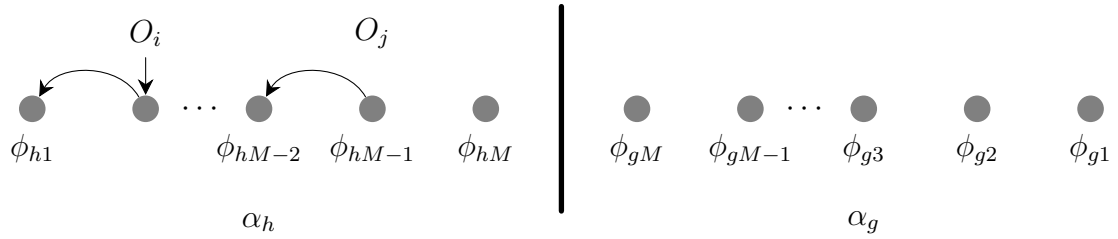
Input:

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

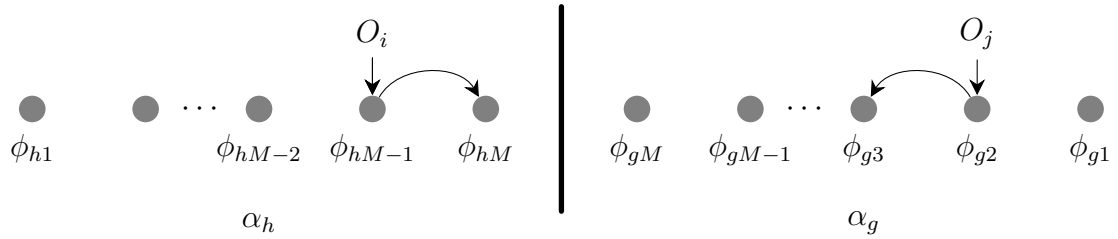
Output:

- The next states of the O_i's.

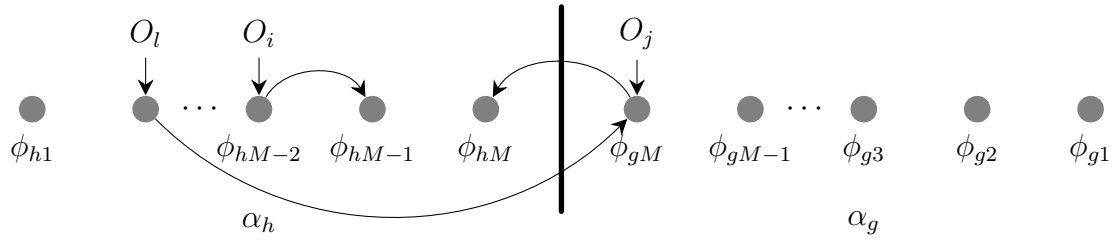
1: **begin**
2: **if** $\xi_i \bmod N \neq 0 \wedge \xi_j \bmod N \neq 0$ **then** ▷ Both are in internal states
3: $\xi_i = \xi_i + 1$
4: $\xi_j = \xi_j + 1$
5: **else if** $\xi_i \bmod N \neq 0$ **then** ▷ O_i is at internal state
6: $\xi_i = \xi_i + 1$
7: **else if** $\xi_j \bmod N \neq 0$ **then** ▷ O_j is at internal state
8: $\xi_j = \xi_j + 1$
9: **else** ▷ Both are in boundary states
10: $temp = \xi_i$ ▷ Store the state of O_i
11: $\xi_i = \xi_j$ ▷ Move O_i to the same group as O_j
12: $l =$ index of an *unaccessed* object in group of O_j closest to the boundary
13: $\xi_l = temp$ ▷ Move O_l to the old state of O_i
14: **end if**
15: **end**



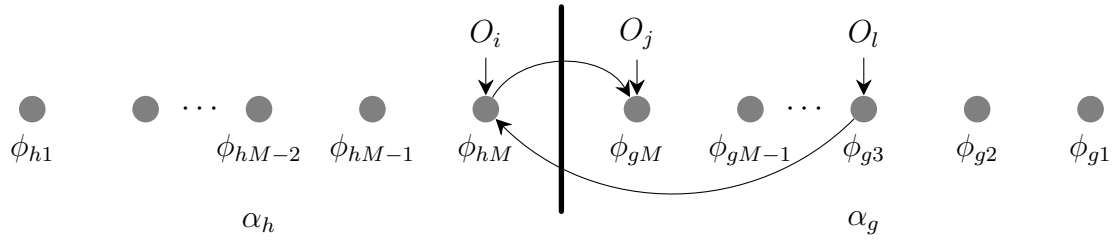
(a) On reward: Move the accessed abstract objects $\langle O_i, O_j \rangle$ towards the extreme states.



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

3.6 Performance of the OMA

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 [83]. From the experimental results, the authors of [83] claim that the scheme is ϵ-optimal.

Experimentally, they also 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 are identical to the ones used in [83]. 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 [83]. 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.

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

In order to assess 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 the $\mathbb{E}$. 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.

Every query presented to the OMA by the $\mathbb{E}$ consists of two objects. 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 3.3. 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). Figure 3.3 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 Figure 3.3, 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 possible states. This step is depicted at the bottom of Figure 3.3, and Ω_0 indicates the initial state of the OMA. At every iteration, a pair given by the $\mathbb{E}$ 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 the Ω^*.

3.6.2 Convergence Criteria for the OMA

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

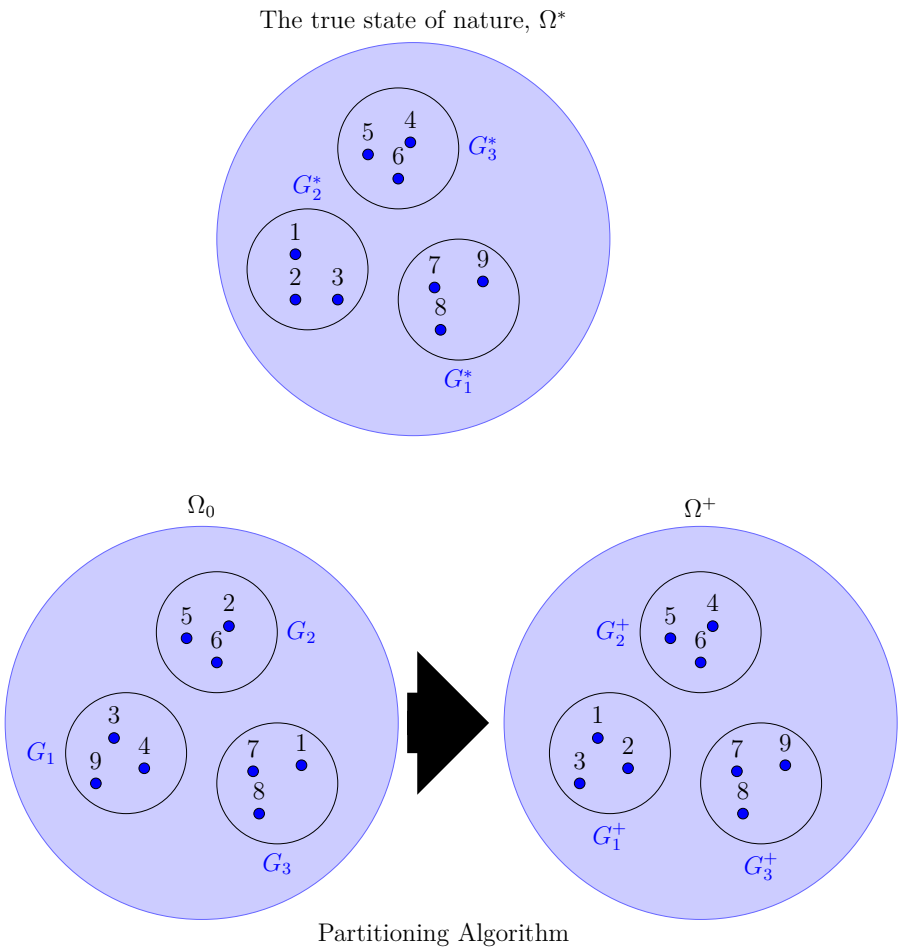


Figure 3.3: A figure describing the partitioning of the objects.

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[7] for the OMA are given in Table 3.1. 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 3.1 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

[7]In our current implementation, we have compared the original OMA algorithm described in [83] with our own simulation results. These are the results shown in Table 3.1. These results, obtained after averaging over 100 independent experiments, are consistent with those given in [83].

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.

Table 3.1: 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.

The convergence of the OMA with regard to time is also very interesting. This is depicted in Figure 3.4 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 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 3.5.

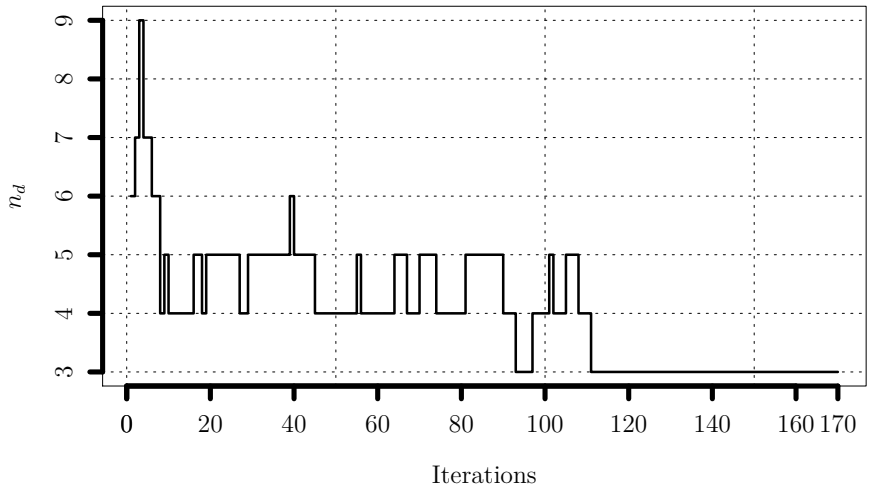


Figure 3.4: A plot of n_d, the number of objects in groups different from their true underlying partitions for the OMA, as the number of iterations (i.e., query pairs) proceed in a single run. Here, $W = 9$ and $R = 3$.

3.7 Applications of the OMA

Due to the poor convergence of prior OPP/EPP solutions, as mentioned earlier, theywere never utilized in real-life application domains. However, since 1986, when the first paper on the OMA was published by Oommen *et al.* [84], it has been applied to variety of real-life problems and domains. It has led to new and improved techniques to address a number of emerging challenges in the field of machine learning, all of which require the task of partitioning in one way or another. These areas include database systems, image retrieval, distributed computing, graph theory, constraint

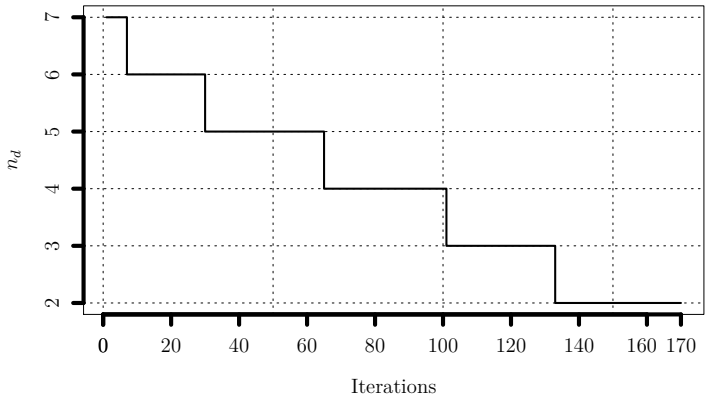


Figure 3.5: A plot of n_d, the number of objects in groups different from their true underlying partitions for the OMA, as the number of iterations (i.e., query pairs) proceed in an ensemble of 100 runs. Here, $W = 9$ and $R = 3$.

satisfaction problems, cryptanalysis, reputation systems, and many other NP-hard problems. This subsection briefly surveys the solutions proposed in these areas.

Soon after proposing their initial OMA solution, Oommen and Ma, in [82], devised an alternate object partitioning method which uses a Stochastic LA. Their solution converged an order of magnitude faster than the best known algorithm in the literature. This machine, called the Stochastic Migration Algorithm (SMA), was even used to solve the general OPP, although its accuracy was limited.

Database Systems: One of the first applications of the OPP was proposed for the design and implementation of Database Systems. Consider a database in which there are W Attributes stored in R Relations. In the interest of simplicity, we assume that R is determined *a priori*. The question is now one of knowing which attributes are to be associated with the specific Relations. Clearly, if two Attributes are often accessed together, it is advantageous that they reside in the same Relation so that whenever one is accessed the other is simultaneously accessed if their common Relation is fetched. By designing the database in such a manner the number of "Join" operations required to respond to a query can be significantly minimized. Thus, in this setting, if the

database administrator opts to include $\frac{W}{R}$ Attributes in each Relation, the solution to the EPP can lead to the understanding of which Attributes should be associated with each other, and thus be co-located in the same Relation.

Keyboard Optimization: The Keyboard Optimization Problem involves assigning multiple characters on a keyboard to minimize the number of collisions. This, for example, can be used to determine the best ordering of the characters {'a', ..., 'z'} on the telephone keypad. One can easily see that this can be reduced to the OPP where the characters are the objects and the partitioning determine the sets they fall into. Oommen *et al.* in [86] and [87], solved this problem in a fast and efficient manner and with a minimum number of ambiguities[8]. The solution that they proposed used the OMA as a primitive component.

Image Retrieval: Consider a database of images which are stored in various subdirectories. Typically, these are stored as an overall collection of items. The Image Examination and Retrieval Problem involves gathering the images which are conceptually similar into the same clusters. Thus, without any verbal description of the images, a solution to the problem would automatically gather the images involving for example, planes, cars, sports etc. into their different categories. This could involve a scenario in which a user would be able to browse the images in the database, and retrieve the desired image by using his/her subjective discrimination of how similar it is to the other images offered by the system. The intention would be that the system would automatically re-cluster the images based on these subjective queries. In [79] and [80], the authors utilized the OMA to efficiently solve the Image Examination and Retrieval Problem.

Cryptanalysis: Substitution-based cryptanalysis involves breaking a substitution cipher by observing the plaintext and the corresponding ciphertext. This should not involve the knowledge of the encryption key, but rather only a statistical understanding of the plaintext and the ciphertext corpuses. With a little insight, one can see that this problem is also easily mapped to the OPP. Indeed, it involves mapping a

[8]Their paper actually proposed communications using the telephone keypad a few years before it became a commercial concept. Together with Zgierski [90], he also later proposed a genetic algorithm to solve this problem.

particular ciphertext character onto a plaintext character, so as to align the respective statistical properties. The authors of [88] and [89] used the OMA to solve the Substitution Cryptanalysis Problem without attempting to discover the private key.

Graph Partitioning:The graph partitioning problem involves partitioning of the nodes (vertices) of a graph in such a way the weights of the edges within each subcluster are maximized and the weights between the clusters are minimized. This is a typical scenario in which the nodes of the graph could be conceptual data centers and where the intra-center traffic is maximized while the inter-center traffic is minimized. Graph partitioning has numerous other applications cited [77] and [78]. The problem is known to be NP-hard. With regard to the OMA, the authors of [77] and [78] proposed the first reported LA-based solution, which also outperformed other state-of-the-art solutions, including Tabu-search methods.

The algorithm proposed by the authors of [78] selected an edge at random and if the two ending vertices belonged to the same partition it will be penalized. The edge was rewarded if it belonged to a different partition. Oommen and de St. Croix further suggested that they could employ an auxiliary algorithm such as the one due to Kernighan and Lin [47] to rapidly find an approximate solution as a starting point for the LA-based solution.

Parallel and Distributed Process Mapping: The OMA has found multiple applications in the fields of parallel and distributed computing. Consider the case where we have W processes which are to be assigned to R processors. In its simplest form, this is referred to as the mapping problem. In this setting, the processes which are assigned to the same processors need minimal communication, while the processes assigned to distinct processors will require significant inter-processor data transfer. Problems of this sort also appear in grid and cloud computing, and can be mapped onto complex combinatorial and integer optimization problems, and was earlier solved using traditional heuristic approaches. In the Static Mapping Problem (SMP), one assumes that the *order* in which the processes are completed is not so crucial, but rather that the task of assigning the processes to the processors so as to minimize the overall communication cost, is more critical. The problem requires some pre-knowledge of the various processes and their interaction. Usually, this information

is not known *apriori*, but it is learned during the planning phases. In a pioneering work, the OMA was applied to solve the SMP [36] and [37].

Multi-Constraint Static Mapping: The OMA has also been successfully applied to solve a *Multi*-constraint Static Mapping Problem, which falls within the family of integer-based Combinatorial Optimization problems. The discrete nature of the solution space renders problems within this category to be extremely hard due to the vast number of possible configurations within the solution space, and due to the fact that there is no simple way of exploring it except by an exhaustive search. Indeed, in most cases, finding the optimal solution is intractable, and thus most of the solutions are heuristic-based. Again, as in the OPP, the problem consists of assigning a set of objects into mutually exclusive classes (where the objects which are "similar" are placed into the same class), except that in this case, the criteria for grouping them have multiple assignment-based conflicting constraints. Horn *et al.*, in [38] and [39], presented the first reported solution to grouping these objects together in such a multi-constraint fashion, where the constrains could be complementary or even contradictory. They proposed a futuristic approach by invoking a *stochastic weak estimator*-based criterion, and by thereafter applying the specific digraph features of the relations between the elements (i.e., the processes/processors) to achieve the mapping. Horn *et al.* also came up with the first known fully decentralized OMA-based solution to the assignment problem involving multiple constraints in [38] and [40].

Adaptive Data Structures: A traditional method for retrieving data from a list of elements is to perform a linear or a sequential search. When the list size is large and the queries obey an unknown probabilistic distribution, search methods do not perform well if the list is static. The recognized way to address this issue is by rendering the list to be "self-organizing" so that it dynamically adapts itself to the distribution of the queries without involving an estimation phase, so as to minimize the search cost. Typical self-organizing operations involve the *Move-to-Front* and *Transposition* rules. In [5] however, the authors proposed an OMA-based solution which gracefully divides the list into sublists and uses the concept of *Lists_on_Lists* (LOLs). This is, again, achieved without estimating the access probabilities. The LOL method

involved multiple layers of self-organization. Whenever a query was received, a linear search was done to locate the element searched for. It was then moved within *its* sublist by instrumenting one of the self-organizing rules. Additionally, the sublist *itself* was moved within the list of sublists by resolving to another higher-level self-organizing rule. The hardest question encountered was that of determining which elements were to reside in the various sublists, and this was accomplished by using an OMA to partition the elements into groups. This method was shown to yield a significantly lower search cost, and it outperformed the best known algorithms within this problem domain.

Secure Statistical Databases: Another application of the OMA involved the security of statistical databases. These databases included the statistical information associated with individuals and enterprises. One such solution is referred to, in the literature, as "Microaggregation". Microaggregation can be seen to be a clustering mechanism. The goal of Microaggregation is to gather a set of records into clusters with some constraints on the size of the groups and, usually, one assumes that there is a proximity measure of interest involved in this process. The Micro-Aggregation Problem (MAP), involves assigning a set of individual records in a microdata file into a number of mutually exclusive and exhaustive groups. In a multivariate setting, this problem is NP-hard. Many heuristic methods has been proposed to solve this problem, and these include the principles of hierarchical clustering, graph-theoretic clustering, genetic algorithms and fuzzy clustering. All these heuristics attempt to minimize the Information Loss (IL) encountered by reproducing the database only in terms of its clusters. A contradiction arises here because of the Statistical Disclosure Control (SDC), which seeks an equilibrium between confidentiality and the data utility criteria. This renders the task of enforcing the minimum IL in this problem to be cumbersome. The majority of the recent and pertinent research focuses on *reducing* the IL value as much as possible. The novel contributions of the research proposed in [26] and [27] was an OMA-based solution which could find the best trade-off between the IL and the SDC. The authors of [26] and [27] proposed and applied the first known OMA-based solution to the MAP, and this was significantly superior to the other state-of-the-art methods.

Distributed Databases: Distributed Database Systems (DDSs) are another emerging application of the OMA. A DDS involves a graph of sites connected together through a communication medium with each site owing its own database. The sites collectively share the databases so that a user can have access to data anywhere in the network, thus permitting him to effectively access a "local" database. In such a scenario, data fragmentation and location are two major factors involved in defining the data integrity and in minimizing the access time respectively. The database designer seeks to minimize the fragmentation and yet store the data in the overall database so as to achieve the minimum access time. This advocates the need for an algorithmic method to allocate the fragments properly so as to minimize the access time required for an incoming query. The data allocation problem is NP-Hard, and due to the application domain, one requires fast (and possibly, randomized) heuristics to operate efficiently and acceptably. The authors in [60], proposed an OMA-based method to solve this problem. It also out-performed many of its contemporary algorithms with a lower standard deviation. This was an indication of it being an algorithm possessing a higher stability in different run-time scenarios.

Reputation Systems: A Reputation System (RS) provides recommendations to users of the Web by extracting information from an aggregate of user-provided feedback responses about the quality of a certain service. Most of the available solutions to this problem in the literature, are generic and prone to "ballot stuffing"and "bad-mouthing". Often the reviews are written by "un-happy" buyers while a "satisfied" costumer rarely spends time, and often spends no time at all, to provide feedback responses. This, clearly, leads to unfair ratings of a product. This phenomenon can impact the degree of *trustworthiness* of the performance of the RS. Unreliable reviews negatively influence the commercial and enterprise values of fair business practices as they impact the success of reliable businesses. In a recent article, the authors of [130] formulated a unique solution that utilized the OMA, independent of prior knowledge, to resolve the above-mentioned issues. The proposed solution can operate in an unknown stochastic Environment, yielding a near-optimal solution by means of a fast-converging and accurate LA-based methodology that operates at a low computational cost. This method gradually captured the characteristics and the identity of

the users who provided fair ratings, and also of those who were unfair in rating products in the system, even if the users made unintentional mistakes. The OMA-based method was proven to efficiently function under any degree of unfair binary ratings, and at the same time, it was able to behave adaptively if the users' trustworthiness changed over time.

Cloud Computing: OMA-based applications can be found in the area of cloud computing as well. The graph partitioning algorithm [77] described earlier, was applied by the authors of [43] to address the common issue encountered by the majority of data centers, namely, that of experiencing a high traffic load to be distributed among a set of Virtual Machines (VMs). The approach traced the data-communication traffic using an OMA-based paradigm, and simultaneously invoked a simulated annealing-based algorithm for assigning the resulting clusters to the server racks. The improvements obtained through the application of this method was significant; it was far more superior to the other contemporary solutions reported in the literature. The proposed method yielded an overall advantage of 90% reduction in performance cost with stable results, along with the simultaneous growth in the intracluster-traffic and a decrease in the intercluster-traffic. Another issue in VMs is the associated cost and efficiency of live migrations in cloud services. A recent research by the authors of [120] addressed these issues so as to decrease the migration time and increase the efficiency. This approach was able to group all the "talkative" VMs and to migrate them in parallel, and this, in turn, led to the reduction in the amount of inter-group traffic. The results reported demonstrated an optimization of over 99% in the case-studies when compared to a sequential migration.

3.8 Modeling the Environment for the EPP

To enable us to consider how we can further improve the OMA, it is educative to first develop a formal model for the Environment that generates the queries from which the partitioning is to be learned.

3.8.1 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[9] as:

$$\begin{aligned}\mu^*_{i,j} = &\ P(R_k) \cdot P(A_j|A_i) \cdot P(A_i), \forall i,j \text{ if } \langle A_i, A_j \rangle \in R_K, \text{ with } k \in \{1, \cdots, R\}, \\ = &\ 0 \text{ otherwise,}\end{aligned} \tag{3.1}$$

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 (3.1) 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[10] of the form:

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}^*_1 & \underline{0} & \dots & \underline{0} \\ \underline{0} & \mathcal{M}^*_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{0} & \cdots & \dots & \mathcal{M}^*_R \end{bmatrix} \tag{3.2}$$

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[11]. 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.

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

[10] 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.

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

Theorem 3.1. *The matrix $\mathcal{M}_r^*, (1 \leq r \leq R)$, is a matrix of probabilities of size $\frac{W}{R} \times \frac{W}{R}$ possessing the following form:*

$$\mathcal{M}_r^* = \begin{bmatrix} 0 & \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} \\ \frac{R}{W(W-R)} & 0 & \cdots & \frac{R}{W(W-R)} \\ \vdots & & \ddots & \vdots \\ \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} & 0 \end{bmatrix} \tag{3.3}$$

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 $1 \leq i, j \leq \frac{W}{R}$. 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, $2 \leq j \leq \frac{W}{R}$, 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 (3.3), and that the sum of all the probabilities in $\mathcal{M}^*$, given in Equation (3.2), is unity. Hence the result! □

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

3.8.2 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:

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}_1^* & \underline{\theta} & \dots & \underline{\theta} \\ \underline{\theta} & \mathcal{M}_2^* & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{\theta} & \cdots & \dots & \mathcal{M}_R^* \end{bmatrix}, \tag{3.4}$$

where $\underline{\theta}$ and $\mathcal{M}_r^*$s are specified as per Equation (3.5) and Equation (3.6) respectively.

Theorem 3.2. *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* (3.4)*, where:*

$$\underline{\theta} = \theta_o \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & 1 \end{bmatrix}, \tag{3.5}$$

and

$$\mathcal{M}_r^* = \theta_d \cdot \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix}, \tag{3.6}$$

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* (3.8)*.*

Proof. As in the case of Theorem 3.1, 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[12] 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 (3.6) 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 (3.6).

[12]This must be contrasted with Theorem 3.1 where $\mathbb{E}$ cannot choose the second element from any other class.

If A_2 belongs to a class distinct from A_1, as opposed to the case of Theorem 3.1, 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 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 (3.5) 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:

$$\theta_d(\frac{W}{R}-1)+\theta_o(W-\frac{W}{R})=1, \tag{3.7}$$

whence, we can see that θ_d has the form:

$$\theta_d = \frac{1-\theta_o(W-\frac{W}{R})}{\frac{W}{R}-1}. \tag{3.8}$$

Hence the theorem! □

3.9 Extracting Convergent Information from $\mathbb{E}$

From the presentation of the pursuit algorithm in Chapter 2 (and also of the OMA principle), one would have got a sense that they had been thoroughly investigated from the early days of their inception. However, on a closer inspection, we realize that there are many unexplored avenues.

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 [29] 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.

Based on concepts akin to the unrelated field of feature selection, there are two candidate methods to achieve the task of extracting convergent information from $\mathbb{E}$. These are listed below as :

- *Filter*-based approaches,
- *Embedded* approaches.

We shall later show that these principals can be used without alluding to the methods proposed in the feature selection domain. We now consider the case when we invoke a filtering approach, and we shall consider *Embedded* methods later.

3.9.1 Filtering $\mathbb{E}$ by Using the Pursuit Concept

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.

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 the $\mathbb{E}$, 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 [84].

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 the $\mathbb{E}$'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, which we have earlier visited in Chapter 2, is quite simply the following: Inexpensive estimates[13] 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.

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

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

threshold, we opt to treat the query in question as a divergent query – in the spirit of the pursuit paradigm.

3.10 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 3.8.1 and 3.8.2 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.

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 3.1 and 3.2, our task is to estimate the corresponding entries in $\mathcal{M}_i^*$ in Equation (3.4). 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 amd 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 3.6 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.

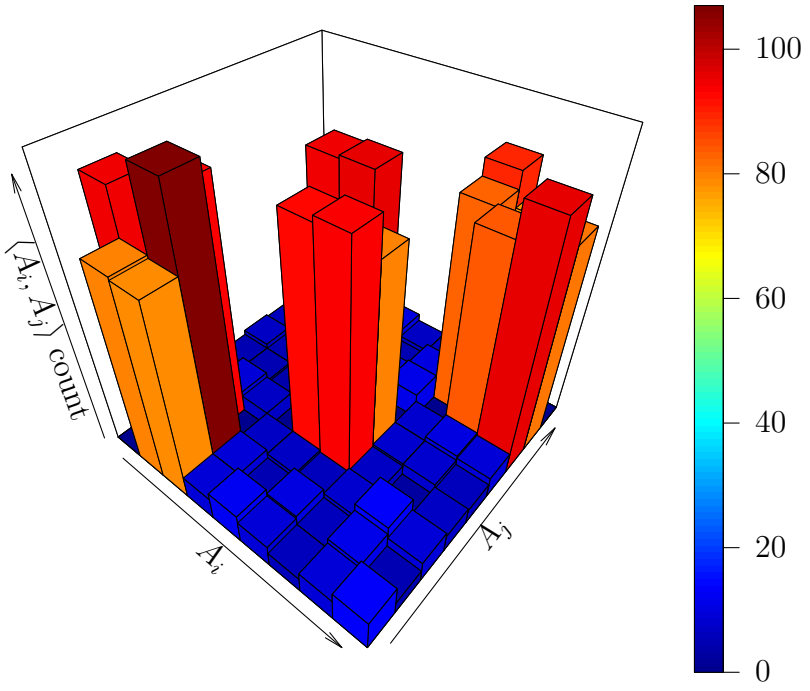


Figure 3.6: The estimation of $\mathcal{M}^*$ which allows us to invoke the pursuit concept. The number of partitions, R, in this case, is 3 and $p = 0.8$.

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 3 and 4 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 $\mathcal{P}[A_i, A_j] > \tau$ 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.

3.11 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.

3.11.1 Data Generation

The approach which is used for generating the query pairs is identical to what was discussed earlier in Section 3.6.1. 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

Algorithm 5 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 $\{\langle A_i, A_j\rangle, \text{ where } i \neq j\}$.
- 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 T queries **do**
    Read query $\langle A_i, A_j\rangle$
    $\mathcal{Z}[A_i, A_j] = \mathcal{Z}[A_i, A_j] + 1$
    $\mathcal{Z}[A_j, A_i] = \mathcal{Z}[A_i, A_j]$
    $\mathcal{P}[A_i, A_j] = \dfrac{\mathcal{Z}[A_i, A_j]}{\sum_{k,l=1}^{W} Z[A_k, A_l]}$ ▷ 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

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 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 (3.4). 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 Figure 3.3.

3.11.2 Setting the Threshold Value

As we discussed in Section 3.10, 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[14].

- **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 3.1 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 3.2, 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 (3.5). 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

[14]In some cases, in the experimental results reported in Table 3.2, 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.

objects appearing from different groups, the estimates will converge to θ_o if the model satisfies the condition of Theorem 3.2. Thus, if we succeed in setting the threshold value τ to be marginally greater than θ_o, 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 (3.6). Thus, if the conditions of Theorem 3.2 are satisfied, the estimates of the entries will all be 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.

3.12 Simulation Results and Discussion

In this section we study the effectiveness and convergence speed of the POMA by evaluating and by comparing our results with those presented in [29] and those reported in Table 3.1 for various values of R and W.

3.12.1 Simulation Results

In our experiments, the number of states in every action was set to 10, and the algorithm was considered to have "converged" as soon as all the objects in the POMA fell within the last two internal states[15]. An identical methodology presented in

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

Section 3.11.2 was followed to populate the parameters of the POMA and the results are summarized in Table 3.2.

In the experiments, the threshold τ was set to be 0.01 for the first two rows of the table, and to 0.02 for the rest of the rows. Also, the number of iterations used to estimate the value of τ was 10 for rows 1 and 2, 60 for rows 3, 4 and 5, and 120 for the last row. The simulation results are based on an ensemble of 100 runs with different uncertainty values, (i.e., values of p) ranging from $0.7-0.9$. One observes that for any 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 scheme utilizes the Reward/Penalty functions.

A comparative discussion of the experimental convergence results of the POMA and the OMA is given in the next section.

3.12.2 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 3.1 and 3.2, 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 3.1 and 3.2.

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 are identical to the one used in [83]. 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

Table 3.2: 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.

BAM, we refer the reader to [83]. 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 convergence of the OMA with regard to time is also very interesting. This is depicted in Figure 3.4 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 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 3.5.

It is worth noting that the results of Table 3.2 does not take into account any of the improvements proposed by the authors of [29]. By merely using the OMA and the pursuit information, we can, indeed, achieve superior results. The advantage gain 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 Chapter 4.

In order to evaluate the effectiveness and convergence speed of the POMA, we have compared our results with those presented in [29] and those reported in Table 3.1 for various values of R and W. In our experiments, the number of states in every action

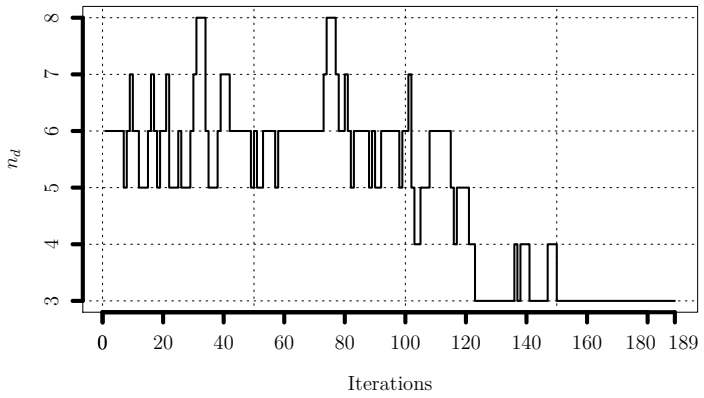


Figure 3.7: A plot of n_d, the number of objects in groups different from their true underlying partitions for the POMA, as the number of iterations (i.e., query pairs) proceed in a single run. Here, $W = 9$ and $R = 3$.

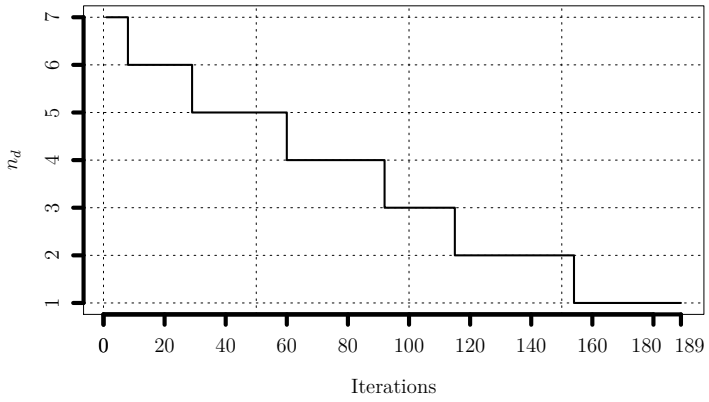


Figure 3.8: A plot of n_d, the number of objects in groups different from their true underlying partitions for the POMA, as the number of iterations (i.e., query pairs) proceed in an ensemble of 100 runs. Here, $W = 9$ and $R = 3$.

was set to 10, and the convergence was expected to have taken place as soon as POMA fell within the last two internal states[16]. The approach explained in Section 3.11.2 was followed quite precisely with regard to setting the parameters of the POMA. The results obtained are quite amazing and summarized in Table 3.2. 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 $0.7 - 0.9$.

When we consider the execution of the algorithm, we observe that the argument presented in Theorem 3.2 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.

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, 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

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

pursuit matrix, the computational performance is identical to that of the OMA. In spite of this, we obtain results that are two to three times superior. In our opinion, this is quite remarkable.

3.13 Conclusion

In this chapter, 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 was provided as a benchmark.

With regard to generating the query stream, we 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. Also, 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. 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 the next chapter we shall show how the pursuit concept can be used to enhance the partitioning technique proposed by Gale *et. al* [29].

"People do not decide to become extraordinary. They decide to accomplish extraordinary things."

Sir Edmund Hillary

4

Enhancing the OMA: State and Pursuit Principles

4.1 Introduction

In an *ideal Environment*, under certain conditions, there is a chance that the OMA gets "trapped" in a *deadlock* situation which prevents it from converging to the correct partitioning. This was alluded to earlier. Although, an ideal environment is very desirable, its appearance in real-life applications is impossible. However, in near-optimal environments (for example, when $p = 0.9$), such a deadlock situation can lead to a very slow convergence rate which intensifies as the complexity of the problem increases[1].

[1]The novel results reported in this chapter, obtained from this research endeavor, have been published in [104], and its extended journal version is currently being revised for publication [105].

In Section 4.2, we will discuss the original improvements proposed by Gale *et al.* in [29] to address this deadlock scenario. In Section 4.3 we will describe, in detail, the *Enhanced OMA* (EOMA) method that they proposed. The experimental results that compare the EOMA with the previous solutions are given in Section 4.4. The major contribution of this chapter is then proposed in Section 4.5, where we incorporate the pursuit phenomenon into the EOMA to obtain the best-known version of the OMA in the literature, namely what we refer to as the *Pursuit Enhanced OMA* (PEOMA). We study and analyze the behavior of the PEOMA in Section 4.5.1. The empirical results obtained for the PEOMA are presented in Section 4.6. In Section 4.7 we discuss the different aspects of the PEOMA that demonstrate its superiority over the previously-proposed methods, and suggest further research directions that can be explored. Section 4.8 concludes the chapter.

4.2 Enhancing the Original OMA

Although the original OMA was a pioneering algorithm that had significant applications, like any other algorithm, it has some limitations. First of all, the OMA algorithm ignores the quality of the input provided by the environment. Indeed, it has no mechanism to enhance its performance by processing the queries based on an inference of 'p'. Secondly, the swapping of the objects between the classes is not always necessarily achieved in the the best possible manner. This is because, when the number of actions is large, the probability of getting multiple subsequent rewards for the same action becomes very small. We now discuss how these drawbacks can be rectified.

More specifically the original OMA algorithm proposed in [83], has two drawbacks, namely:

1. A slow convergence rate, and
2. A sensitivity to noise in the environment (i.e., where we encounter a difficult $\mathbb{E}$).

The authors of [29] proposed three modifications to improve the performance of the original algorithm. We list them below.

1. **Initial Distribution of Objects**:
   Instead of initially distributing the objects between the $R \times N$ states randomly, the authors of [29] suggested that all of the objects be placed in the boundary states of the respective classes[2]. Since, in the early iterations of the algorithm, the OMA-based partitioning is, typically random, it is desirable to swap objects, between the groups, as early as possible. By positioning the objects in the boundary states, we reduce the average number of iterations it would take for the OMA to exchange the objects between the groups.

2. **Breaking the Deadlock**:
   If a pair of objects $\langle O_i, O_j \rangle$ are accessed together and one of them, say, O_i, is in the boundary state, while the other is in a non-boundary state (Case **(c)** in Figure 3.2), the authors of [29] recommended that the updating rule be split into two sub-cases listed as follows:

   a) If there is an object in the boundary state of the group containing O_j, O_i is swapped with such an object in the boundary state of O_j's group.

   b) Otherwise, if there is no object in the same boundary state of O_j, the rule is identical to the one proposed in [83].

3. **Redefining Internal States**:
   The OMA's internal states are susceptible to the environment's noise. This decelerates the convergence to the most internal states. In fact, if an object is in the most internal state after several rewards, we would like to deter it from being taken out of the "converged state" by the unfortunate occurrence of a *single* faulty query. To rectify this situation, the authors in [29] proposed to re-define the most internal states of the OMA to be the last two innermost states for every class, rather than it being only the *single* last state. These would, for example, be the states ϕ_{g1} and ϕ_{g2} in the class α_g in Figure 3.2.

[2]This was, indeed, the way the experiments in [83] were conducted, we see that although it was not clearly communicated in the paper.

4.3 Designing the EOMA

The learning process of an enhanced LA, as proposed by Gale *et al.*, is based on the same principles explained in Chapter 2 and 3, and leads to what we shall refer to as the Enhanced OMA (EOMA). The EOMA has all the properties of the original OMA. It is an automaton with N states and R actions. The objects are initially randomly distributed among the boundary states. It also manipulates the physical objects, and like the OMA, it operates on the abstract objects. The 1-to-1 correspondence between the physical and the abstract objects $\{O_1, O_2, \cdots, O_W\}$ remains the same. Gale *et al.* in [29] , introduced three enhancements to the original OMA to improve its partitioning efficiency and increase its convergence speed[3]. The enhancements are found in the initialization of the OMA, its structure, and in an enhancement in the penalty policy. We first review the initialization and structural modifications, and later study the change to the penalty policy:

1. **Initial Boundary State Distribution**: All of the objects are initially distributed at the respective boundary states of their respective classes. Thus, all the objects can migrate to appropriate classes on the first query that accesses them instead of being queried multiple times before reaching the boundary state.

2. **Redefinition of Internal States**: The internal states of the OMA are vulnerable to the divergent queries. If an object is in the internal state of its class, a divergent query can easily displace it from this internal state. In [29], Gale *et al.* diminished the vulnerability of the OMA to divergent queries by redefining the internal state to include "the two innermost states of each class", rather than the single innermost state. The effects of divergent queries are lessened, although once an object reaches the internal state, it essentially, ignores all subsequent convergent queries. They also proposed that an even more superior solution would be to have an unbounded number states for each class. Thus, an object would be considered to be in an "internal" state if its state index is

[3]The description about how the original OMA simulation was carried out is given in Chapter 3.

greater than a predefined integer, m.

With respect to the modification to the penalty policy, consider an optimal environment, $\mathbb{E}$, in which $p = 1$. We now argue that in such an environment, there is a possibility that the original OMA cannot converge to a correct partitioning in such an environment[4]. Similarly, if the environment is near optimal ($p \approx 0.9$), depending on the states of the objects and the queries given by $\mathbb{E}$, it would take the OMA a large number of iterations to converge to the correct partitioning. We demonstrate this for an ideal environment below.

When a query pair $\langle A_i, A_j \rangle$ is given by $\mathbb{E}$, the LA displaces the abstract objects based on the respective states occupied by O_i and O_j. Consider the scenario when we have two sets of objects, $\mathcal{A} = \{A_1, A_2, A_3\}$ and $\mathcal{B} = \{B_1, B_2, B_3\}$, and where the following access query sequence appears repeatedly:

$\{\langle A_1, A_2 \rangle, \langle A_1, A_3 \rangle, \langle A_2, A_3 \rangle, \langle B_1, B_2 \rangle, \langle B_1, B_3, \langle B_2, B_3 \rangle \rangle\}$,

and is presented by $\mathbb{E}$. Let us assume that the objects are in the internal and boundary states of the OMA, as below:

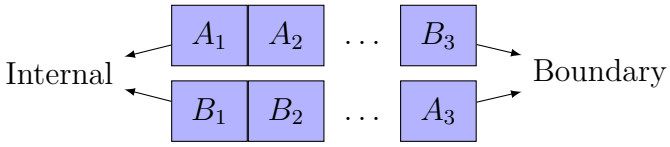


Figure 4.1: The initial distribution of the objects which could lead to a deadlock situation.

The reader can easily see that after processing the six queries presented by $\mathbb{E}$, all the abstract objects will return to be at the exact same positions. Thus, the algorithm becomes trapped in a *deadlock* and can never attain to the correct partitioning, since neither the As nor the Bs can migrate to the respective boundary states in order to be swapped and moved to the same class. The reader will see that such a scenario can be generalized for $k > 3$ objects with $\mathcal{A} = \{A_1, A_2, \cdots, A_k\}$ and $\mathcal{B} = \{B_1, B_2, \cdots, B_k\}$,

[4]Of course, such an ideal Environment seldom occurs. However, in every real-life scenario (for example in a database application where two attributes in a specific relation are almost always accessed together), such a deadlock will *always* occur.

and where the analogous sequence of query pairs arrive, as stated above. Using the same notation, the positioning of the abstract objects is as below:

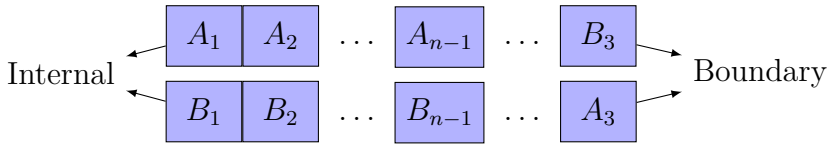


Figure 4.2: A generalized distribution of the objects that could lead to a deadlock situation.

We again see that the deadlock condition will not be broken by an appropriate sequence of query pairs.

Breaking the Deadlock: Give a query pair of objects $\langle O_i, O_j \rangle$, let us assume that O_i, is in the boundary state, and O_j is in a non-boundary (internal) state of another class. If there exists an object in the boundary state of the same class, we propose that it gets swapped with the boundary object O_j to bring both of the queried objects together in the same class. Simultaneously, a non-boundary object has to be moved toward the boundary state of its class. Otherwise, if there is no object in the boundary state of the class that contained O_j, the algorithm performs identically to what was dictated by the OMA. This can be expressed as the following:

> *If one of the objects, say O_i, is already in the boundary state of its class, and the other object, O_j, is in a non-boundary state of its class, if there is an object in the boundary state of O_j's class, that object has to be swapped with O_i, to bring O_i and O_j in the same class and O_j is advanced one state closer to the boundary state of its class. Otherwise, if the boundary state of O_j's class is empty, the algorithm is identical to the OMA's.*

The algorithmic representation of the EOMA and its *reward* and *penalty* procedures are given in Algorithms 6, 7 and 8. Note that Algorithm 7 is identical to Algorithm 3 in Section 3.5 for the OMA.

Algorithm 6 The EOMA Algorithm

Input:

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

Output:

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

1: **begin**
2: Initialization of $\{\xi_p\}$ ▷ This is described in the text
3: **for** a sequence of T queries **do**
4: Read query $\langle A_i, A_j \rangle$
5: **if** $\xi_i \ N = \xi_j$ div N **then** ▷ The partitioning is rewarded
6: Call ProcessRewardEOMA($\{\xi_p\}, A_i, A_j$)
7: **else** ▷ The partitioning is penalized
8: Call ProcessPenaltyEOMA($\{\xi_p\}, A_i, A_j$)
9: **end if**
10: **end for**
11: Print out the partitions based on the states $\{\xi_i\}$
12: **end**

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

Input:

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

Output:

- The next states of the O_i's.

1: **begin**
2: **if** $\xi_i \bmod N \neq 1$ **then** ▷ Move O_i towards the internal state
3: $\xi_i = \xi_i - 1$
4: **end if**
5: **if** $\xi_j \bmod N \neq 1$ **then** ▷ Move O_j towards the internal state
6: $\xi_j = \xi_j - 1$
7: **end if**
8: **end**

Algorithm 8 *ProcessPenaltyEOMA*$(\{\xi_p\}, A_i, A_j)$

Input:

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

Output:

- The next states of the O_i's.

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

If one compares Algorithm 8 to Algorithm 4 in Section 3.5, he sees how the penalty function has been modified for the boundary cases so to reflect the *deadlock breaking* policy.

4.4 Results of the EOMA

The simulation results obtained by introducing all of the three above-mentioned modifications are given in Table 4.1. In this table, we have reported the results from various Environments, with probabilities $p = 0.9, 0.8$ and 0.7, where $p = 0.9$ is the near-optimal Environment. Such an Environment is easy to learn from. On the other hand, for the case where $p = 0.7$, we encounter a difficult-to-learn scenario. Our results have also compared our own implementation of the OMA algorithm described in [29] with the EOMA's simulation results. These are the results displayed in Table 4.1. All the simulations reported were done on an ensemble of 100 experiments to guarantee statistically stable results.

From the table we clearly see that as the value of p increases, the queries are more informative, and the convergence occurs at a faster rate. As before, the complexity of the classification problem has two criteria which are both observable in the tables, i.e., the number of objects in every group, $\frac{W}{R}$, and the number of groups, R. As the number of objects and groups encountered increase, the problem becomes increasingly complex to solve.

By way of example, consider the entries in Table 4.1 for the case when $R = 3$ and $W = 18$, and when the probability of receiving a paired query was $p = 0.7$. It took the EOMA, on average, 857 iterations to converge to the correct partitioning. For the same experimental setting, it took the POMA $1,881$ queries, while the original OMA attained full convergence to the correct partitioning in $2,111$ iterations. The advantage of using these three enhancements is thus evident.

The convergence of the EOMA with respect to time is displayed in Figure 4.3 for the case where $W = 9$ and $R = 3$. As before, at any given time slice the figure shows the number of objects that are not correctly placed with regard to the true underlying partition at any given time slice. As expected, the algorithm starts with a

Table 4.1: Experimental results for the *Enhanced* OMA (EOMA) done for an ensemble of 100 runs.

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

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

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

$EOMAp\mathcal{X}$**:** $\mathcal{X}$ refers to the Environment's probability of generating samples within the same class.

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

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

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

large number of objects which are located in random partitions. This number steadily decreases with time to a very small value. As in the case of Figure 4.3, the graph is not monotonic for any given experiment. But from the perspective of an ensemble, the performance is much more monotonic in behavior. This can be seen in Figure 4.4.

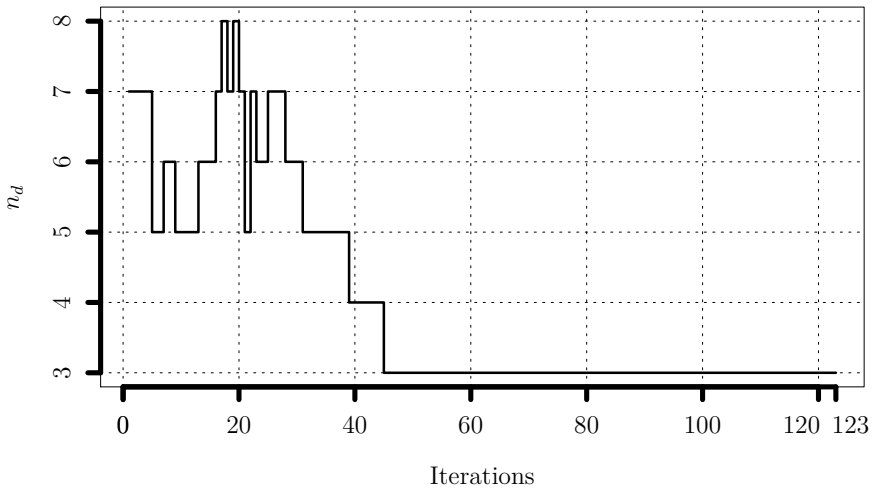


Figure 4.3: A plot of n_d, the number of objects in groups different from their true underlying partitions for the EOMA, as the number of iterations (i.e., query pairs) proceed in a single run. Here, $W = 9$ and $R = 3$.

4.5 Enhancing EOMA with a Pursuit Paradigm

As mentioned earlier, the goal of this chapter is to propose and test how the pursuit concept can be used to further optimize the EOMA. As a brief reminder, we refresh the reader's memory by observing that an analogous attempt was the main thrust of Chapter 3, i.e., to optimize the virgin form of the OMA by invoking pursuit principles. We now consider how this goal can be attained with the EOMA as the basic framework.

The methodology by which we incorporated the pursuit concept into the OMA required us to formally model noise-free and noisy queries in Sections 3.8.1 and 3.8.2. However, this is the precise dilemma that the EOMA faces. On the one hand, it would

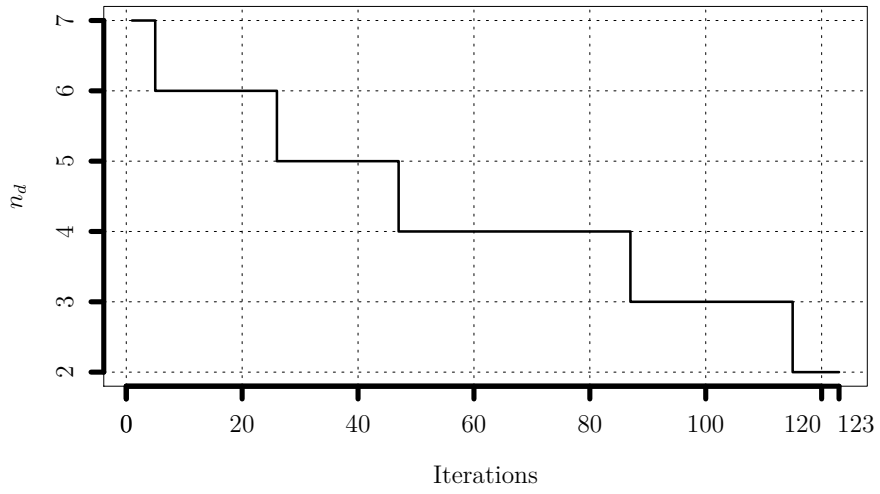


Figure 4.4: A plot of n_d, the number of objects in groups different from their true underlying partitions for the EOMA, as the number of iterations (i.e., query pairs) proceed in an ensemble of 100 runs. Here, $W = 9$ and $R = 3$.

be advantageous, from a partitioning perspective, to have a noise-free Environment. However, as we have explained in Section 4.3, it is precisely such noise-free Environments that lead to deadlock situations. Consequently, an attempt to elevate a noisy Environment to become a noise-free Environment would only defeat the purpose by exaggerating deadlock scenarios. With this as a background, we can observe that from an OMA's perspective, a pursuit concept must not be applied so as to the OMA make the Environment noise-free.

Rather than seeking to transform the Environment to be noise-free, we shall see how the pursuit framework can be introduced to filter out queries, in a manner similar to that which was introduced in Chapter 3. This, as before, relies on the models of the Environment for noise-free and noisy queries, explained in Sections 3.8.1 and 3.8.2 respectively, and these discussions will not be repeated here. However, rather than seeking to make the Environment noise-free, we will again accept or reject queries from the incoming stream. To accomplish this, we again apply the same paradigm introduced in Chapter 3, explained below for this specific setting.

4.5.1 How the PEOMA is Designed

To design the PEOMA, as before, we attempt to incorporate the pursuit principle by estimating the Environment's reward/penalty probabilities. This could, of course, be done based on either a ML or Bayesian methodology. As these estimates become more accurate, we force the LA to converge to the superior actions at the faster rate.

In this setting, as in the POMA, the estimation will serve to achieve filtering. By obtaining estimates of the pair-wise query probabilities, we will be able to filter out the less informative ones. Observe that this has the effect of making the Environment to be noise-free, and although this could create deadlocks, the EOMA, by nature of its design, will be capable of resolving them.

This implies that we will have to maintain a $W \times W$ array of values, where every entry in the matrix provides an estimate of the joint access for the corresponding pair. This yields the degree of "certainty" of a query pair being proposed by $\mathbb{E}$, and if the estimate of the probability of this event is less than a predefined threshold, we will treat is a "divergent" query.

Considering the models of the noisy and noise-free Environments presented in Sections 3.8.1 and 3.8.2 respectively, we know that the information about the convergent/divergent queries resides in the estimation matrix $\mathcal{M}^*$, defined in Theorems 3.1 and 3.2 which are derived by the traditional ML method. As the number of queries processed increases, the entries in the block-diagonal matrices will converge to correspond to the true underlying partitions of a noise-free Environment, and by utilizing a user-defined threshold, τ, which is reasonably close to zero, the divergent queries can be ignored.

The resolution of the dilemma, that we eluded to above, does not lie in "hands of" the OMA, but rather by invoking the EOMA. Thus, even though the queries appear to arrive from a noise-free Environment, the consequent deadlock situations will be processed by the EOMA, and quickly force the corresponding abstract objects to exit the deadlock. Observe that the chance of "bad" inputs affecting the LA is also valid for the POMA.

The overall conclusions of the above discussion are the following:

- The stream of queries is processed using an *estimation* phase;
- The divergent queries are filtered using a *thresholding* phase, that serves as a filter for the above estimates;
- The deadlock scenarios are resolved using the enhancements of the EOMA over the OMA;
- The convergence criterion of making the two most internal states of every group to report convergence, makes the entire process converge even faster.

The formal algorithm for this is given in Algorithm 9.

4.6 Experimental Results: PEOMA

In this section, we compare the performance of the EOMA with its pursuit counterpart, the PEOMA. We first, discuss the initial steps necessary to run the experiments. We then explain, in some details, about the initial requirements of such a pursuit paradigm. Next, we report the experimental results obtained by simulating the PEOMA in different Environments. Finally, we analyze and discuss the results obtained and close the chapter after an overall summery.

4.6.1 Data Generation

In order to achieve a meaningful comparison between the various schemes, we utilize the same data generation method used in Sections 3.6, 3.11 and 4.4 for the OMA, POMA and the EOMA respectively. The sequence of operations is done so as to obtain every query pair based on a predefined distribution, where as before, we assume that there are W objects, and R groups, and the Environment, $\mathbb{E}$, is characterized by a probability p which suggests pairs of objects from the same group as being queried together.

The first object of every query is drawn uniformly at random from among the W existing objects. This means, the probability of selecting any given object is $\frac{1}{W}$. All

Algorithm 9 PEOMA algorithm

Input:

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

Output:

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

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

the R groups are equally-likely, and will be selected with a probability of $\frac{1}{R}$ as done earlier in Section 3.11. We face two scenarios for choosing the second element:

- **To choose the second object from the same group**: Here, also as before, we enforce the condition that we choose an object from among the $W-1$ remaining objects, except that it must belong to the same class as the first-selected object. The process is described earlier and is not given here to avoid repetition.
- **To choose the second element from a different group**: In this case the second object is selected from among the $R-1$ remaining groups, and once the group is selected, the object is choosen uniformly from *that* group. This too is achieved as described in the earlier sections.

The above-described scheme for generating the query pairs was invoked so as to obtain a continuous stream of queries, while simultaneously maintaining the Environment's probabilistic integrity. Clearly, the goal of the PEOMA is to converge to a final partitioning as depicted in Figure 3.3, and with a convergence speed that, hopefully exceeds that of the previously-recorded algorithms.

4.6.2 Setting the Threshold Value

The PEOMA and the previously-described POMA essentially use the same parameters κ and τ. As in the POMA, selecting the values for the threshold, τ, is crucial. This is a user-defined value set by the operator who has to ascertain whether a certain query is to be considered as a divergent query or not. Indeed, the value of τ is very crucial for the PEOMA to not only converge but to also operate in a meaningful manner. It turns out that even though this filters the Environment to become noise-free, all the previously-discussed factors that affect the value of τ will again be pertinent. The same is true when it concerns setting the value for κ, the number of iterations needed for the estimates of the Pursuit matrix to converge. Since the methods by which these parameters are set are identical to what is described for the POMA in (as reported in Section 3.10), to avoid repetition.

4.6.3 Simulation Results for the PEOMA

In this section, we report the simulation results that inform us of the effectiveness and convergence rate of the PEOMA, and compare its performance with the results presented in [29] and those reported in Table 4.1 for various values of R and W and in different Environments. The number of states in every action was set to be a constant, 10, as in Section 4.3. The convergence condition was also identical to the one specified in Section 4.3, and was assumed to have taken place as soon as all the objects in the PEOMA fell within the last two internal states. Further, the query probability approximations were updated after receiving every single query.

The parameters used to run the PEOMA were also identical to what was described Section 3.11.2. With no prior knowledge of the query probabilities available, an educated guess for initializing τ is less than $\frac{1}{W}$, and an initial number of iterations required to obtain an estimate of τ is $\left[\left(\frac{W}{R}\right)^2 - \frac{W}{R} \right] \times R$.

The simulation results based on an ensemble of 100 runs are given in Table 4.2 for various uncertainty values for p, ranging from $0.7 - 0.9$. We should mention that the performance significance of the PEOMA is, really, not noticeable for easy problems where we had a small number of objects and groups, and where the noise level is low. But this becomes invaluable when we encounter a large number of actions as well as a stream of divergent queries (when p is "smaller") throughout the simulation, especially if we factor in the number of iterations used to obtain an estimate of τ. Indeed, the results are so remarkable; they are unrivaled by any other variation of the OMA reported in the literature till the date of writing this.

By utilizing both the internal improvements as well as the pursuit concept in extracting information from the Environment, the PEOMA is capable of operating in extremely complex (highly noisy) Environments, and solving difficult problems rapidly. Indeed, in the case where the Environment's noise is high and there are many objects to be grouped, the OMA and the EOMA require an extremely large number of queries to converge to the correct partitioning when compared to the PEOMA.

A comparative discussion of the PEOMA with the POMA and the EOMA is given in the next section.

4.7 Discussion

When we monitor the execution of the algorithm, we observe that the argument presented in Theorem 3.2 becomes quite pertinent 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 PEOMA's accept/reject policy. Indeed, for a pair given by $\mathbb{E}$ at time t, if the corresponding estimate for the query was characterized by a value less than τ, the query is treated as a divergent pair. Otherwise the pair is considered to contain valuable information, and the PEOMA's Reward/Penalty functions are invoked.

The enhancements introduced in [29] are all structural. By using a quantitative approach, the pursuit paradigm is, essentially, further capable of realizing the quality of the incoming pairs and providing the LA with a higher chance to receive and process the "filtered", more accurate, inputs. In contrast with the internal enhancements, the pursuit paradigm's focus is about understanding the Environment's characteristics. Thus incorporating these two methods in conjunction with each other, has led to superior results, and the experimental results speak for themselves.

By way of example, if the PEOMA is compared with the original EOMA, (proposed by Gale *et al.* in [29]), one can see that the EOMA can solve the partitioning problem with $p = 0.9$ and 6 groups with 2 objects in each group, in 102 iterations. For the same problem, the PEOMA required only 97 iterations to converge. For a difficult-to-learn Environment ($p = 0.7$) and a more complex partitioning problem with 18 objects in 3 groups, the EOMA needed 857 iterations to converge. The PEOMA required only 472 iterations to converge, which is more than *1.8 times* better than the EOMA, 7.5 better than the POMA, and more than *ten times* better than the original OMA.

The convergence graph for a single experiment of the PEOMA is depicted in Figure 4.5. This considers the same case that was studied earlier, in which $W = 9$ and $R = 3$. The figure depicts the number of objects that are not correctly grouped together with regard to the true underlying partition at any given time instant. The

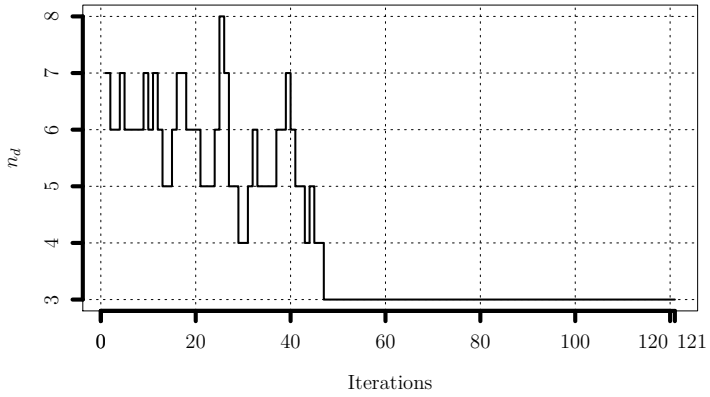


Figure 4.5: A plot of n_d, the number of objects in groups different from their true underlying partitions for the PEOMA, as the number of iterations (i.e., query pairs) proceed in a single run. Here, $W = 9$ and $R = 3$.

PEOMA, similar to its predecessors, starts with a large number of objects in the wrong partitions, and steadily decreases this index to smaller values. This behavior is portrayed in Figure 4.5. However, if one takes the ensemble average of this metric over 100 experiments the performance is much more monotonic in behavior. This is clear from Figure 4.6. The reader should observe the considerable performance that is gained by a very little additional computational cost. Again, by comparing Tables 4.1 and 4.2, although the gain is not significant for simple problems and easy Environments, it becomes remarkably high for complex partitioning experiments.

4.8 Conclusion

In this chapter, we reviewed the Enhanced version of the OMA (EOMA) proposed in [29], and considered an un-addressed issue, namely that of encountering a large number of actions and a high level of noise, and the task of optimizing it by utilizing the pursuit concept. This led to the *Pursuit Enhanced OMA*, PEOMA, which could solve the Equi-Partitioning Problem (EPP). We also compared it to the best known

Table 4.2: Experimental results for the PEOMA approach done for an ensemble of 100 runs.

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

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

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

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

N: Number of objects to be partitioned.

W/R: Number of objects in every class.

R: Number of classes in the partitioning problem.

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

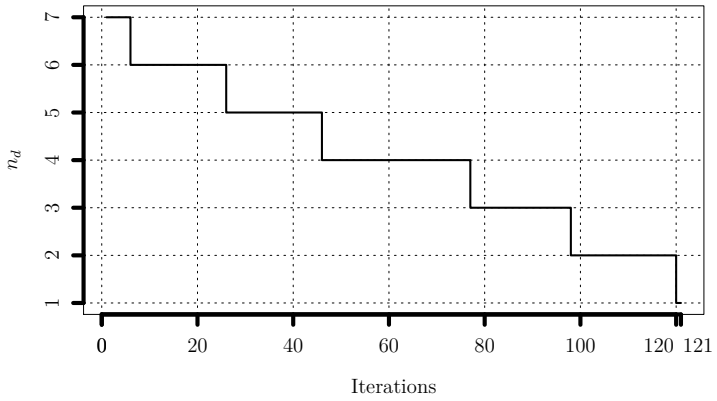


Figure 4.6: A plot of n_d, the number of objects in groups different from their true underlying partitions for the PEOMA, as the number of iterations (i.e., query pairs) proceed in an ensemble of 100 runs. Here, $W = 9$ and $R = 3$.

versions of the OMA.

The PEOMA utilized the exact same pursuit principles used in the POMA in Chapter 4. We argued that the so-called "deadlock" situation in addition to the "divergent" queries, would be able to drastically slow down or even, in some rare cases, stop the convergence of the OMA. Consequently, we considered how the deadlock problem could be resolved. We then combined it with the Pursuit concept that was used as an effective accept/reject policy, which in turn, led to a significant increase in the performance of the original OMA. This was also able to take care of the scenarios when we had a large number of groups and a very noisy Environment.

The experimental results that compare the PEOMA with the EOMA were submitted in Section 4.6.3. We were able to show how the PEOMA out-performed the EOMA. By studying a spectrum of problems, which extend from easy-to-learn to difficult Environments, and for simple or complex partitioning problems, we realized that the PEOMA's performance can be almost *two* times faster than the EOMA. But if we compare the results with the original OMA, the immense performance gain leads to up to *forty* times less iterations required for a complete convergence—which is by

no means insignificant. It is fascinating to note that the reduction in the number of iterations required by the PEOMA can be seen to be a consequence of a pursuit-like filtering phase in all problem domains.

In summary, the introduced modifications enhance the OMA's performance without any significant computational overhead in all the examined Environments, even when the partitioning problem is inherently challenging.

"If the enemy of my enemy is my friend, then a friend of my enemy, is my enemy."

Pyreglide

5

Transitive Pursuit Enhanced OMA

5.1 Introduction

The OPP paradigms, which we have studied in the last few chapters, would be less difficult if the uncertain components associated with the problem were absent. In reality, its complexity stems from the uncertainty in the underlying phenomena, and the fact that we are dealing with random Environments renders the problem to be far more complex. The goal of any solution to the OPP, and in particular of the OMA paradigm, is that *it should be able to determine the quality of the inferences of the objects that are associated with each other*. In other words, it would be advantageous to infer the "*trustworthiness*" of the Environment that provides information about any specific object and about the objects that it "wants" to be associated with[1]. This

[1]The novel results reported in this chapter have been accepted for publication in the journal *IEEE Access* [107].

is, actually far from trivial, because as the designers of the algorithms, we cannot "control" the input that we receive.

As a preface, we mention that the results of the previous chapters demonstrate that since the OMA uses the principles of LA, and invokes a reward/penalty policy-based scheme, its performance degrades drastically as the number of partitions increases and/or as the Environment becomes more noisy. This is due to the fact that the probability of receiving rewarding pairs for the partitions to converge decreases, and/or the probability of receiving diverging pairs causes a more sluggish convergence. In Chapters 3 and 4, we have tried to mitigate this phenomenon by integrating the Pursuit paradigm that has powerfully enhanced the field of of LA.

In Chapter 3 we demonstrated that invoking and incorporating the information in the pursuit matrix increases the performance significantly. The first result, which is reported in Chapter 3, utilizes the *pursuit* method, which estimates the statistics of the Environment. By using these estimated values, one could infer whether to accept or reject the incoming query pairs. We were also able to successfully incorporate this paradigm to the Enhanced OMA in Chapter 4. The results were orders of magnitude faster than their non-pursuit versions. As far as we know, this represented the state-of-the-art when the paper [103] was published.

Although the pursuit paradigm has been quite successful in identifying so-called divergent queries, one observes that it utilizes the reward/penalty policies in a passive way. Whenever the queries are unavailable, or whenever they are not in the pipeline to be processed, it waits for the input from the Environment, and thereafter invokes the proper policy based on the pursuit matrix. The inevitable consequences of such a policy-enforcing mechanism are:

- If the number of objects or actions are large, the events triggering the reward and/or penalty become scarce;
- When operating in a noisy Environment, the algorithm is dormant whenever it waits for a query from the Environment;
- As the size of the partitioning problem increases, the probability of an object receiving a request for an update decreases, hindering the overall convergence

rate of the algorithm.

The primary motivation for this chapter is to address the issues mentioned in the above observations. We shall show that the pursuit matrix is not only useful to achieve query identification. Rather, that it can also provide meaningful inferences from the observed query stream to yield a policy that *actively* (as opposed to passively, or in a dormant manner), engages reward/penalty operations. This is accomplished by incorporating an estimation of a measure that expresses the similarity/relation between numerous objects that have *not* been accessed. We are not aware of any similar strategy that has been used either in the field of LA or in the area of partitioning.

Although the pursuit paradigm proposed in Chapters 3 and 4 has improved the OMA's performance significantly, it only incorporates the knowledge in the most basic or primitive form, i.e., the pairwise relationship, which is established between *two* objects. It is noteworthy that although a query pair can occur in two ways, say $\langle A_i, A_j \rangle$ or $\langle A_j, A_i \rangle$, since the labeling of the objects is arbitrary, we do not take the pains to distinguish between the first or the second form. The analysis of this relation consists of statistical modeling, specifying whether certain pair of objects tend to occur together, which has been the core of the proposed method in Chapter 4.

The observation mentioned above, i.e., $\langle A_i, A_j \rangle \implies \langle A_j, A_i \rangle$ is, clearly, what describes a reflexive relation. Since we do not distinguish between either of these two occurrences in the pair, we implicitly assume such a reflexivity. However, there is another property of the pursuit matrix that has neither been discovered nor been utilized in partitioning-based problems. This is the property of *transitivity* which is the central "asset" which we shall take advantage of. After the initial transient stage of the algorithm, as the pursuit matrix converges, we shall show that the property of transitivity becomes enforced. Thus, if $\langle A_i, A_j \rangle$ and $\langle A_j, A_k \rangle$ are related, the pair $\langle A_i, A_k \rangle$ can be inferred. Consequently, in this research, we propose a novel solution which takes advantage of a measure by which objects in larger k-tuples (not just pairs) in a partition tend to cluster together by merely investigating pairwise convergence properties. This can enhance the performance of the consequent OMA-based algorithms.

The implication of this is significant. Indeed, even though the pair of objects

$\langle A_i, A_k \rangle$ is not accessed together, we can infer that they should be together in the same group, because of the elements that they are *already associated with*. We will argue that this transforms the dormant nature of the algorithm. It also enhances the speed significantly. The details of all these concepts will be provided in the body of this chapter.

The results obtained on benchmark Environments demonstrate that our current transitive versions is nearly two times faster than the non-transitive versions. The current version is nearly 90 *times* faster than the original OMA [84] for certain Environments. By all metrics, these results are incredible.

5.1.1 Chapter Organization

In 5.1 we initiated our discussion by first describing how we can improve the Pursuit principle . Section 5.2 provides the theory and the necessary background which are used later in Section 5.4. In Section 5.3, we discuss how the Environment can be modeled and we provide a probabilistic representation for it. Section 5.4 incorporates the idea of transitivity and introduces the concept of *Inferred* queries as opposed to the *Real* queries presented by the Environment. In Section 5.4.1, the convergence conditions of the Transitive PEOMA (TPEOMA) are discussed in detail. Section 5.5 presents the simulation results for the TPEOMA, after which we study its performance. Section 5.7 concludes the chapter.

5.2 Cohesiveness within Objects in the EPP

The first issue that we encounter when we want to advance the field of resolving the EPP is to see if we can use new criteria to identify which objects belong to the same partition. We intend to investigate how this can be inferred without considering the issues that have been analyzed earlier. It is easy to see that all the objects within an underlying partition should be strongly and directly related to each other, and that they should frequently co-appear in the queries. Such structural patterns

are, in turn, based on so-called casual propositions which should lead towards relational "interactions" between the objects themselves. This is the avenue that we now investigate.

Structural relations that are imposed by the Environment can orient the objects towards a uniformity when there is an "interaction" between a pair of objects. Such relations may be "transmitted" through intermediaries even when two objects are not explicitly examined at any given time instant. This interconnection is directly associated with the relational bonds that these objects possess. We shall now investigate whether this property, which already relates *subgroups* and not just pairs, can be quantified by various specific properties that can be extracted from the Environment.

From Figure 3.3 and $\mathbb{E}$'s *modus operandus*, we see that there are four common phenomena that each subgroup possesses:

1. The frequency of objects co-occurring;
2. The relative frequency of the objects in a pair belonging to distinct partitions;
3. The symmetric property of the queries in any pair presented by $\mathbb{E}$;
4. The reachability of the objects in a partition within the graph representing the set of all objects.

Our task is to consider how information about all these issues can be extracted from the Environment. As one observes, the first two entries above, namely, those dealing with the frequencies of the pairs, constitutes the principles motivating the pursuit-based solutions [103] and [105], where these frequencies were obtained by employing a ML scheme. Our task now is to incorporate the latter two entries. We formalize the symmetric property, as seen in the partitioning problem:

Definition 5.1. A binary relation $\mathcal{R}$ over a set of objects W is symmetric if

$$\forall O_i, O_j \in W : O_i \mathcal{R} O_j \iff O_j \mathcal{R} O_i. \tag{5.1}$$

Theorem 5.1. *The model of* $\mathbb{E}$ *and the solution invoked by any pursuit-based paradigm of the EPP possess the property of symmetry.*

Proof. Our first task is to show that the model of $\mathbb{E}$, as discussed above, possesses symmetry. To do this, consider the probability of $\mathbb{E}$ presenting a query pair $\langle O_i, O_j \rangle$. This means that the first element O_i is chosen from any group with probability $\frac{1}{W}$. For the sake of simplicity, let this group be G_r. Consider now the scenario where the second element is from the same group. In such a case, the probability of choosing O_j from the same group is $p \cdot \frac{R}{W-R}$, because, p is the probability of $\mathbb{E}$ choosing an element from the same group, and $\frac{1}{W/R-1} = \frac{R}{W-R}$ is the probability of choosing any element other than O_i. The product of these two quantities yields the probability $P(\langle O_i, O_j \rangle) = \frac{1}{W} \cdot p \cdot \frac{R}{W-R}$. Similarly, if O_j is chosen first, as in the pair $\langle O_j, O_i \rangle$, the probability of choosing O_j will be $\frac{1}{W}$, after which O_i will be chosen, yielding $P(\langle O_j, O_i \rangle) = \frac{1}{W} \cdot p \cdot \frac{R}{W-R}$. The symmetry is clear because these two expressions are identical.

For the other scenario, the second object in $\langle O_i, O_j \rangle$ is chosen from a different group other than the group of O_i. The corresponding probability for O_i remains to be $\frac{1}{W}$. When it concerns choosing the second element O_j, the probability of choosing this element from a different group is given by $(1-p) \cdot \frac{1}{W/R} \cdot \frac{1}{(R-1)}$. Here, $(1-p)$ is the probability of choosing O_j from a different group, i.e., $P(O_j \notin G_r)$, and once the decision is made to choose from another group, we observe that there are $(R-1)$ equally likely groups of size $\frac{W}{R}$ each. By applying a similar principle, the probability for the case $P(\langle O_j, O_i \rangle)$ can be shown to be $\frac{1}{W} \cdot (1-p) \cdot \frac{R}{W(R-1)}$. Again, the probabilities for $\langle O_i, O_j \rangle$ and $\langle O_j, O_i \rangle$ are identical, implying that $\mathbb{E}$ possesses the symmetry property.

Now that we have seen that $\mathbb{E}$ possesses symmetry, we want to show that both the pursuit-based algorithms, the POMA and PEOMA, are able to infer this symmetry. Without belaboring the point, we note that both these algorithms compute and maintain the so-called pursuit matrices. As explained in [103, 105], this matrix consists of a sequence of blocks, and within the blocks along the diagonal, the groups naturally fall into clusters. The entire pursuit matrix consists of estimates of the probabilities of the objects being accessed together which is thus a symmetric matrix. The principle behind the POMA and PEOMA is that if an estimate is less than κ, we ignore the corresponding query. It is thus clear that if $\langle O_i, O_j \rangle$ is ignored because it

is considered to be a divergent query, the pair $\langle O_j, O_i \rangle$ will also be a divergent query. Thus, both the POMA and PEOMA infer and use the property of symmetry.

While this property was not specifically mentioned in [84, 29], it was tacit, and implemented in [103] and [105]. The details of the corresponding modified OMA algorithms for both these scenarios were also presented there. □

We now consider the property of transitivity as it appears in the partitioning problem. We can formalize this property as follows:

Definition 5.2. A binary relation $\mathcal{R}$ over a set of objects W is transitive if:

$$\forall O_i, O_j, O_k \in W : (O_i \mathcal{R} O_j \wedge O_j \mathcal{R} O_k) \implies O_i \mathcal{R} O_k. \tag{5.2}$$

Theorem 5.2. *The model of $\mathbb{E}$ proposed for the EPP possesses the property of transitivity from a probabilistic perspective.*

Proof. Consider any three objects O_i, O_j and O_k. For any probability value 'p', that characterizes $\mathbb{E}$, we want to show that if the probability of $\langle O_i, O_j \rangle$ and $\langle O_j, O_k \rangle$ being generated is some quantity λ, the probability of $\langle O_i, O_k \rangle$ being generated is also precisely λ.

From the arguments presented in Theorem 5.1, one observes that the probability of O_i and O_j belonging to the same class is $P(\langle O_i, O_j \rangle) = \frac{1}{W} \cdot p \cdot \frac{R}{W-R}$. Similarly, the probability of O_i and O_k belonging to the same class is $P(\langle O_i, O_k \rangle) = \frac{1}{W} \cdot p \cdot \frac{R}{W-R}$.

Consider now the probability $P(\langle O_i, O_k \rangle | \langle O_i, O_j \rangle \wedge \langle O_j, O_k \rangle)$. By virtue of the independence of the queries, this has the form:

$$\begin{aligned} P(\langle O_i, O_k \rangle | \langle O_i, O_j \rangle \wedge \langle O_j, O_k \rangle) &= \frac{P(\langle O_i, O_k \rangle) \cdot P(\langle O_i, O_j \rangle) \cdot P(\langle O_j, O_k \rangle)}{P(\langle O_i, O_j \rangle) \cdot P(\langle O_j, O_k \rangle)} \\ &= P(\langle O_i, O_k \rangle) \\ &= \frac{1}{W} \cdot p \cdot \frac{R}{W-R} \end{aligned}$$

From this we see that the probability of O_i and O_k belonging to the same class is identical to the probability of O_i and O_j belonging to the same class, and this, in turn, is also identical to the probability of O_j and O_k belonging to the same class. Hence the result. □

Since $\mathbb{E}$ is transitive, our aim is now to have the LA infer this transitivity and to further enhance the PEOMA. We shall proceed to show how this can be achieved in two steps. In the case of a noiseless environment, as in the case of the PEOMA, we will show that the pursuit matrix is composed of block-diagonal matrices. We will see that each of these blocks naturally demonstrates transitivity, and that this therefore can be used to achieve faster convergence. But since a noiseless environment is non-existent, our next task will be to see what happens in the case of real-life noisy environments. Again, we shall show that if the pursuit matrix is appropriately thresholded, the entries become unity and zero, which allows us to demonstrate transitivity and thus, invoke reward/penalty operations even while the environment is dormant and not generating any new queries.

5.2.1 Transitivity for the Noiseless Environment

Let us consider the case when there is the absence of "noise" in the Environment, i.e., there is absolute certainty. We can infer the actual value of the relation between O_i and O_j by a quantity $\mu^*_{i,j}$, expressed[2] as in Equation (3.1).

Thus, $P(R_k)$ 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. The set of values of $\mu^*_{i,j}$ in Equation (3.1) is used to represent the elements of the policy matrix $\mathcal{M}^*$. We shall now examine the properties of $\mathcal{M}^*$ and show that it demonstrates the transitivity phenomenon.

Theorem 5.3. *The matrix $\mathcal{M}^*$ demonstrates the transitivity of $\mathbb{E}$.*

Proof. The proof consists of two parts. The first part, which specifically describes $\mathcal{M}^*$, is rather identical to the corresponding proof found in [103]. The second part shows the transitivity of $\mathbb{E}$.

Since $\mathbb{E}$ uniformly selects a pair of elements from two distinguished groups, the matrix $\mathcal{M}^* = \left[\mu^*_{i,j}\right]$ is *block-diagonal*[3] of the form Equation (3.2).

[2]Please note that this is the *total* probability of $\mathbb{E}$ presenting the pair $\langle O_i, O_j \rangle$.

[3]It can be shown that by an appropriate re-mapping of the indices, one can obtain a block matrix in which the elements of the partitioning Ω^* are adjacent, even if the original matrix is cluttered.

The matrix $\underline{0}$ represents a square matrix containing only 0's. On the other hand, each matrix $\mathcal{M}_r^*, (1 \leq r \leq R)$, is a matrix of probabilities of size $\frac{W}{R} \times \frac{W}{R}$ possessing the form given in Equation (3.3).

This is true since the relevant distributions are uniform. If the first element is denoted by $\mathbb{E}$ is A_i from class G_r, the probability that the second element is selected from any of the other elements in G_r is equally divided between these elements. Likewise, 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 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 $[i, j]$ is the likelihood that $\mathbb{E}$ chooses the objects $\langle O_i, O_j \rangle$ which is introduced as a result of the real-world query of the form $\langle A_i, A_j \rangle$. Clearly, since single objects cannot be accessed alone at any time, $\langle O_i, O_i \rangle$ cannot be a possible pair since the real pair $\langle A_i, A_i \rangle$ is impossible in the real world. Thus, the diagonal elements are all 0.

Consider now the scenario when the first element chosen by $\mathbb{E}$ is O_1. Then the second element can be any one of the O_j's, $2 \leq j \leq \frac{W}{R}$, 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 $P(O_1) = \frac{1}{W}$, the total probability $P(O_1, O_j) = \frac{R}{W(W-R)}$, 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 (3.3), and that the sum of all the probabilities in $\mathcal{M}^*$, given in Equation (3.2), is unity. This concludes the first part of the proof.

Now, to demonstrate the transitivity, we concentrate on three arbitrary indices i, j and k. If i and j do not belong to the same block, then μ_{ij}^* from Equation (3.1) must be 0. On the other hand, if they do belong to the same block, say M_r^*, the corresponding entry in the μ_{ij}^* should be $\frac{R}{W(W-R)}$. Similarly, if we consider the indices j and k, we see that if they do not belong to same block, the entry μ_{jk}^* should be 0,

and it will also have the value of $\frac{R}{W(W-R)}$ if they do belong to the same block. It is now easy to see that if μ^*_{ij} is $\frac{R}{W(W-R)}$ and μ^*_{jk} is $\frac{R}{W(W-R)}$, it constrains i and k to also be in the same block forcing μ^*_{ik} to also be $\frac{R}{W(W-R)}$. The theorem follows. □

We now examine the real-world scenario, when the Environment is noisy.

5.3 The Noisy Environment

Our task is to now confirm the transitivity for the noisy environment. The information which resides in the pursuit matrix, introduced in [103], can be utilized to extract the relationships between objects or even groups of objects. In order to study and extract these relations, a formal model of the Environment which generates the queries, is necessary, and we shall achieve this presently. This will be shown to demonstrate transitivity and it will thus provide the formal definitions which lays the foundation of the *transitive* pursuit method.

By computing a simple Maximum Likelihood (ML) estimate of how frequently every query $\langle O_i, O_j \rangle$ appears, one obtains an estimate of the underlying probabilities of $\mathbb{E}$ in generating every pair combination. 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 off-diagonal blocks. An example of how the estimate of $\mathcal{M}^*$ looks is displayed in the figure on the left in Figure 5.1 for the simple case when we have three block matrices, i.e., when we are dealing with three distinct partitions. The reader will observe that within each block, the estimates of the probabilities are almost the same but the variations are due to the inaccuracies of the estimates. As we increase the number of query pairs processed, the estimate values in each block will tend to approach their asymptotic values. Similarly, outside of these blocks, the asymptotic values will be correspondingly very small, and the magnitude of these entries will be discussed soon.

Consider now the scenario when we render the situation to be binary. To achieve this, we create an augmented matrix $\mathcal{Q}$. The entry $\mathcal{Q}_{ij}$ is set to be unity if $\mathcal{M}^*_{ij}$ is close to $\frac{R}{W(W-R)}$, and is set to 0 when $\mathcal{M}^*_{ij}$ is less than a user-defined threshold, say

τ_t. In such a case, one observes that the matrix $\mathcal{Q}$ asymptotically becomes binary, where the block-diagonal matrices corresponding to the underlying partitions attain the value of unity, and all the off-diagonal matrices become $\underline{0}$. This is depicted in the figure on the right of Figure 5.1. The transitivity of $\mathbb{E}$ can thus be easily inferred, as we shall do in Theorem 5.2.

The next subsection formalizes the phenomenon.

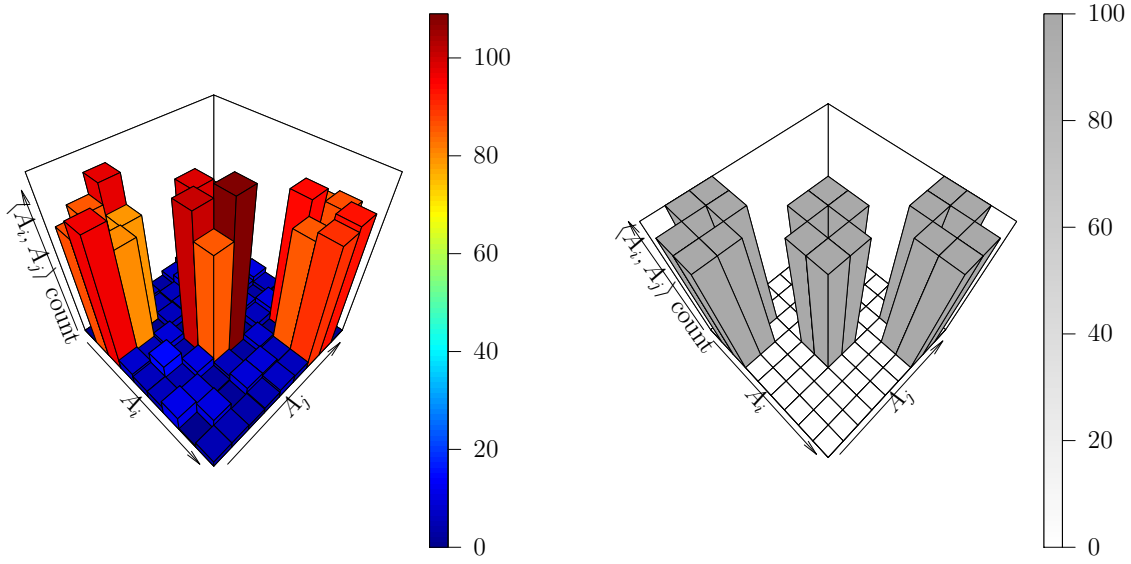


Figure 5.1: The estimation of $\mathcal{M}^*$ which allows us to invoke the pursuit concept. The number of partitions, R, in this case, is 3 and $p = 0.8$. The figure to the left represents the joint probabilities of the objects. The figure to the right is the binary valued representation as in the text.

5.3.1 Modeling the Noisy Environment and its Transitivity

Theorem 5.4. *By a simple thresholding mechanism, the transitivity property of the matrix $\mathcal{M}^*$ remains valid even when the environment is noisy.*

Proof. As in the case of Theorem 5.3, the proof consists of two parts, although the

corresponding parts are more complex. The first part, which specifically describes $\mathcal{M}^*$, is rather identical to the corresponding proof found in [103]. There are a few fine details which are different and these will be highlighted. The second part, which shows the transitivity of $\mathbb{E}$, uses arguments similar to those in the above section.

Since $\mathbb{E}$ uniformly selects a pair of elements from two distinguished groups, the matrix $\mathcal{M}^* = \left[\mu^*_{i,j}\right]$ is *block-diagonal*[4]. In the presence of noise in $\mathbb{E}$, the entries of the pair $\langle O_i, O_j \rangle$ can be selected from two different distinct classes, and hence the matrix, $\mathcal{M}^*$, specifying the probabilities of the accesses of the pairs obeys Equation (3.4). As mentioned in Equation (3.4), $\underline{\theta}$ and $\mathcal{M}^*_r$s are specified as per Equation (3.5) and Equation (3.6) respectively, 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 (3.8).

To prove the above, as in the case of Theorem 5.3, 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 $\langle O_i, O_j \rangle$ 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[5] 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 $\langle O_i, O_j \rangle$ where $2 \leq j \leq \frac{W}{R}$. Since all the $[1, j]$ entries of $\mathcal{M}^*_1$ are equally likely[6], the form of Equation (3.6) 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 (3.6).

If A_2 belongs to a class distinct from A_1, as opposed to the case of Theorem 5.3, the $\underline{\theta}$ matrices in the same block-row with $\mathcal{M}^*_1$ of $\mathcal{M}^*$, cannot be zero matrices anymore.

[4]As in the case of [103] and [105], it can be shown that by an appropriate re-mapping of the indices, one can obtain a block matrix in which the elements of the partitioning Ω^* are adjacent, even if the original matrix is cluttered.

[5]This must be contrasted with Theorem 5.3 where $\mathbb{E}$ cannot choose the second element from any other class.

[6]The equally-likely condition is a necessity to demonstrate transitivity. It was not required in the case of the [103] and [105].

Thus, each entry in the first row outside of $\mathcal{M}_1^*$ must be set to the probability value of O_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 (3.5) 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:

$$\theta_d(\frac{W}{R}-1) + \theta_o(W - \frac{W}{R}) = 1, \tag{5.3}$$

whence, we can see that θ_d has the form:

$$\theta_d = \frac{1 - \theta_o(W - \frac{W}{R})}{\frac{W}{R} - 1}. \tag{5.4}$$

Having described $\mathcal{M}^*$, our next task is to show that $\mathbb{E}$ possesses transitivity. To do this, as explained in the previous section, we maintain an augmented binarized matrix $\mathcal{Q}$. As the number of query pairs increases, the estimate μ_{ij}^* which converged in the noise-free environment to $\frac{R}{W(W-R)}$, would, in the noisy environment converge to θ_d, as given by Equation (5.4). The reader will observe that θ_d in Equation (5.4), is explicitly given in terms of θ_o, and that every element outside the block-diagonal matrix for every group is exactly θ_o. Therefore, if we keep a threshold smaller than θ_o and retain those elements in $\mathcal{Q}$, all the off-diagonal entries in $\mathcal{Q}$ will become 0. Further, all the entries in the block matrices $\mathcal{M}_r^*$ will be $\frac{R}{W-R}$ which can be rendered to be '1' in the binary matrix, $\mathcal{Q}$.

Now, to demonstrate the transitivity, we again concentrate on three arbitrary indices i, j and k. If i and j do not belong to the same block, then the arbitrary

element q^*_{ij} of $\mathcal{Q}$, would be set to 0. On the other hand, if they do belong to the same block, say $\mathcal{Q}^*_r$, the corresponding entry in the q^*_{ij} should be unity. Similarly, if we consider the indices j and k, we see that if they do not belong to same block, the entry q^*_{jk} will be 0, and it will also have the value of unity, if they do belong to the same block. It is now easy to see that if both q^*_{ij} and q^*_{jk} are 1, it constrains i and k to also be in the same block forcing q^*_{ik} to also be unity. The theorem follows. □

As in the POMA and PEOMA, specifying a user-defined threshold close to 0, τ, we will be able to compare every estimate to it, 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 $\langle O_i, O_j \rangle$ should be processed, or quite simply, be ignored. This takes care of the symmetric components of the Pursuit matrix, and also resolves the deadlock. However to consider the transitivity, we now utilize the property asserted in Theorem 5.2, without explicitly maintaining the matrix $\mathcal{Q}$. This is explained in the next section.

5.4 The Transitive PEOMA (TPEOMA)

Our task now is to create an even more enhanced version of the PEOMA, which considers reflexivity and transitivity. We shall divide this in two phases, both of which utilize the pursuit matrix described in [103] and [105]. The first phase is identical to the PEOMA which is given in Algorithm 11. Indeed, before the estimate has converged, we must merely invoke the EOMA 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 EOMA algorithm given in Algorithms 7 and 8 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 $\mathcal{P}[A_i, A_j] > \tau$ 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[7].

The next phase will consider how the property of transitivity can be taken advantage of. The reader will observe that Theorem 5.2 has laid the foundation by which the transitivity concept can be incorporated into the pursuit paradigm. This foundation, of course, relies on inferring the relation between the objects, which can be extracted from the query stream provided by the Environment. This can then be applied to both the noise-free and the noisy environments as presented in previous section. Thus, our goal would be to process the incoming query and, first of all, resolve the issues for the PEOMA, and thereafter, infer the transitive relations between the object pair and the rest of the objects located in the same partition. Again, we shall accomplish this by resorting to the information found in the pursuit matrix and the thresholding described in Theorem 5.2.

Since the pursuit matrix represents the estimates of the likelihoods of objects occurring together, we can utilize this matrix to infer the transitive relations among the objects. Consider a case when the query pair $\langle A_i, A_j \rangle$ is given by $\mathbb{E}$. All the objects that are in a transitive relation with A_i *through* (or rather, because of) A_j are located in the j-th row of the pursuit matrix[8]. The likelihood values among the elements in the j-th row vary depending on the strength of the relations to A_j. We now define a threshold value, say τ_t, which represents the minimum likelihood limit for the elements to quantify the transitive bounds between A_i and the remainder of the objects in the j-th row. Thus, if an element passes this threshold, the algorithm can invoke the the same action for the objects in the transitive relation with A_i. Since this paradigm uses the pursuit matrix to infer the transitivity relation among objects, we refer to this scheme the Transitive PEOMA, TPEOMA. The algorithmic representation of this approach is further explained below.

Up on receiving a reward for a pair of objects in a query by $\mathbb{E}$, the TPEOMA will reward the objects which are in transitivity relation defined by the pursuit matrix and the τ_t value. This behavior can be interpreted as an extrapolation or forecasting of

[7]The reader can refer to [103] for the details on how we can specify the values of τ and κ as far as he PEOMA is concerned.

[8]Since the queries are symmetric, it does not matter whether if we consider row i or j to locate the transitive objects.

what pairs would appear together in the future given by the $\mathbb{E}$. Clearly, in the absence of any a priori knowledge, since we have W objects in total, we can assume that the lowest bound for τ_t should be $\frac{1}{W(W-1)}$. The upper bound for the ideal environment was obtained previously which is $\frac{R}{W(W-R)}$. Thus, $\frac{1}{W(W-1)} < \tau_t < \frac{R}{W(W-R)}$.

In the following section, we discuss how to attain an estimate of τ_t and the necessary conditions required for the convergence of the TPEOMA.

So far the pursuit paradigm has been limited to the interaction between the OMA and the Environment. The accept/reject policy for a query is inferred from the pursuit matrix with a minimum computational overhead. Surprisingly, we can use the pursuit matrix to extract the pairwise relations between the objects in $\mathcal{O}(W)$, where W is the number of objects to be partitioned.

From the above, the reader will observe that regardless of how well the PEOMA performs in a noisy environment or in solving difficult partitioning problems, the overall convergence rate of any OMA-based solution will be a direct function of the query *count* received. Thus, if the number of objects or the partition count increases, it reduces the convergence speed due to the decrease in the probability of receiving a specific converging pair, or receiving enough number of query pairs for each partition so that the PEOMA can actually converge. This is not a design issue of the OMA in general, but rather is an inherent nature of the partitioning problem.

While this phenomenon is true of the PEOMA, a remedy to accelerate the convergence rate of the PEOMA, when the environment is dormant, is to generate so-called *Artificial* queries, and this is precisely what the TPEOMA achieves. Since the transitivity relation determines how objects within a partition are related to each other, if the PEOMA receives a pair of objects from the environment, we can generate a series of *Inferred* queries by determining all of the objects that lie within the transitivity relation with the given query objects. Clearly, these are highly relevant to the original received query even though these *Inferred* queries have not actually occurred. In other words, we predict the future behavior of the environment by incorporating the Pursuit matrix and we can achieve this during the phase when the Environment is "dormant", i.e., when it is generating no queries for the AI system to process. This method is formally given in Algorithm 10 which invokes Algorithm 11, a skeletal

version of the EOMA. Observe that the latter's *Reward* and *Penalty* algorithms are precisely Algorithm 7 and Algorithm 8, and are repeated here as Algorithm 12 and Algorithm 13 merely in the interest of continuity.

5.4.1 Remarks Regarding the TPEOMA

The pursuit matrix collects the likelihood of objects occurring together in the matrix $\mathcal{M}^*$, defined in Theorem 5.3 and Theorem 5.4. In other words it captures the $\mathbb{E}$'s behavior. The TPEOMA seeks a boundary after which the estimate of this likelihood would fall asymptotically.

After enough number of iterations, when the PEOMA is considered to have converged, the entries of the pursuit matrix will quantify the underlying relations between objects by estimating the corresponding probabilities of $\mathbb{E}$. This is can be utilized to infer the underlying reflexive and transitive relations. As the number of queries processed becomes larger, the quantities inside $\mathcal{M}_i^*$ will grow significantly larger than the quantities in each of the off-diagonal blocks in $\mathcal{M}_i^*$. For example, for the case represented in Figure 5.1, $\mathcal{M}^*$ was obtained for the simple case when we have three block matrices, i.e., when we are dealing with three distinct partitions. From the figure, it can be observed that the estimates of the matrix $\mathcal{M}_i^*$ have much higher values for the objects residing in the same group.

The question of when the estimates can be considered as converging is simply determined by a reasonable threshold τ which is obtained after κ iterations. Thus, if $\mathcal{P}[A_i, A_j] > \tau$ 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 obtained values. This concept is demonstrated algorithmically in the TPEOMA algorithm (Algorithm 10), where the reader should spcifically note the following:

1. First of all, the TPEOMA is designed as a standalone unit.

2. While the query statistics are gathered, processed and thresholded in the TPEOMA, the actual migration is achieved by invoking the EOMA (see lines 9, 12 and 15).

However, rather than invoking the entire EOMA, we merely call its skeletal version (SkeletalEOMA) which avoids the input/output operations.

3. In the TPEOMA, if the query is divergent and the pursuit matrix has not converged yet, we merely invoke the EOMA as in line 9. Note that lines 12 and 15 process the convergent queries.

4. With regard to the Skeletal EOMA, we have again written it in a modularized manner, unlike the representation given in [105].

5. Finally, with regard to the *Real* and *Inferred* queries, we invoke the reward/penalty function (line 9 and 12 in Algorithm 10, and lines 6 and 8 in Algorithm 11) for the *Real* queries in the If block (line 8). However, for the *Inferred* queries, we enter the Else block in line 14, and we again achieve this in line 12 of Algorithm 10 and lines 6 and 8 of Algorithm 11. By doing this, we avoid the duplication of code and are also able to treat the *Inferred* queries in the same way as we treated the *Real* queries.

5.5 Simulation Results

This section is dedicated to the simulation results in which we discuss the effectiveness and convergence rate of the TPEOMA, and compare its performance with the results presented and reported in [105] for various values of R and W, and in different Environments. The number of states in every action was set to be a constant, 10, as used in [105] and [104]. The convergence conditions were also identical to the ones specified in [105] and [104], and it was assumed to have taken place as soon as all the objects fell within the last two internal states. Further, the query probability approximations were updated after receiving every single query.

The parameters used to run the TPEOMA are also identical to what was described [105]. With no prior knowledge of the query probabilities available, an educated guess for initializing τ was that it had to be greater than $\tau^* = \frac{1}{W^2-W}$. The expected number

of iterations required to obtain an estimate of τ was that it should have a value greater than $\kappa > \left[\left(\frac{W}{R}\right)^2 - \frac{W}{R} \right] \times R$.

The simulation results are given in Table 5.1 which are based on an ensemble of 100 runs for various uncertainty values for p, ranging from $0.7 - 0.9$. We should mention that the performance significance of the TPEOMA is, really, not noticeable for easy problems where we had a small number of objects and groups, but unlike the PEOMA, it can converge nearly *two* times faster in environments with low levels of noise. This is significant because the difference becomes more noticeable when the difficulty of the problem increases. The TPEOMA also outperforms the PEOMA in solving difficult partitioning problems. Indeed, the results are so remarkable; the *Transitive PEOMA* incorporates the same pursuit matrix to and it is unrivaled by any other reported variation of the OMA.

By utilizing both the internal improvements as well as the pursuit concept in extracting transitivity relations from $\mathbb{E}$, the TPEOMA is capable of operating in extremely complex (highly noisy) environments, and solving difficult problems even faster than PEOMA.

In comparison with the original PEOMA as reported in [105], Algorithm 10 also updates the states of the *inferred* objects which are in the transitivity relation with the received object pair. In summary, the introduced modification enhances the PEOMA's performance by introducing a linear overhead $\mathcal{O}(W)$ by looking in to a row of W objects and choosing the objects which are above τ_t, the transitivity threshold value. This overhead is constant for both cases when the Environment is difficult to learn or the partitioning problem is inherently challenging.

5.6 Discussion

By monitoring the execution of the algorithm, it is obvious that the argument presented in Theorem 5.4 becomes quite pertinent as the total number of divergent query pairs is always much less than the number of convergent pairs. Thus, the value of τ obtained by the steps described in the previous section, can be set as a *threshold* value for the TPEOMA's accept/reject policy.

More specifically, for a pair given by $\mathbb{E}$ at time t, if the corresponding estimate for the query was characterized by a value less than τ, the query is treated as a divergent pair. Otherwise, the pair is considered to contain valuable information, and the PEOMA's Reward/Penalty functions are invoked. By using a quantitative approach, the pursuit paradigm is, essentially, further capable of realizing the quality of the incoming pairs and providing the LA with a higher chance to receive and process the "filtered", more accurate, inputs. In contrast with the internal enhancements, the pursuit paradigm's focus is about understanding the Environment's characteristics.

Although the use of the Pursuit matrix to filter out the divergent queries has been reported earlier [103] and [105], this chapter has taken a bold step to consider the implications of its transitivity properties. The consequence of this property is that we have been able to create *Inferred* queries. In other words, even thought the Environment is unable to provide us with relevant information to the AI algorithm, the transitivity property permits us to "cook up", or rather artificially generate, queries that mimic the Environment's properties. These *Inferred* queries have been utilized in conjunction with the real queries to drastically improve the performance of the learning system.

Thus, by incorporating these two principals in conjunction with each other, has led to superior results, and the experimental results speak for themselves.

By way of example, if the TPEOMA is compared with the previously best-reported algorithm, the PEOMA, (proposed by Shirvani *et al.* in [105]), one can see that the PEOMA can solve the partitioning problem with $p = 0.9$ and 3 groups with 3 objects in each group, in 85 iterations. For the same problem, the TPEOMA required only 65 iterations to converge. For a difficult-to-learn Environment ($p = 0.7$) and a more complex partitioning problem with 18 objects in 3 groups, the PEOMA needed 472 iterations to converge. The TPEOMA required only 244 iterations to converge, which is nearly *two times* better than the PEOMA.

The convergence graph for a single experiment of the TPEOMA is depicted in Figure 5.2. This considers the same case that was studied earlier, in which $W = 9$ and $R = 3$. The figure depicts the number of objects that are not correctly grouped together with regard to the true underlying partition at any given time instant. The

TPEOMA, similar to its predecessors, starts with a large number of objects in the wrong partitions, and steadily decreases this index to smaller values. This behavior is portrayed in Figure 5.2. However, if one takes the ensemble average of this metric over 100 experiments the performance is much more monotonic in behavior. This is clear from Figure 5.3. The reader should observe the considerable performance that is gained by a very little additional computational cost. Again, by comparing Tables 4.2 and 5.1, although the gain is not significant for simple problems and easy Environments, it becomes remarkably high for complex partitioning experiments.

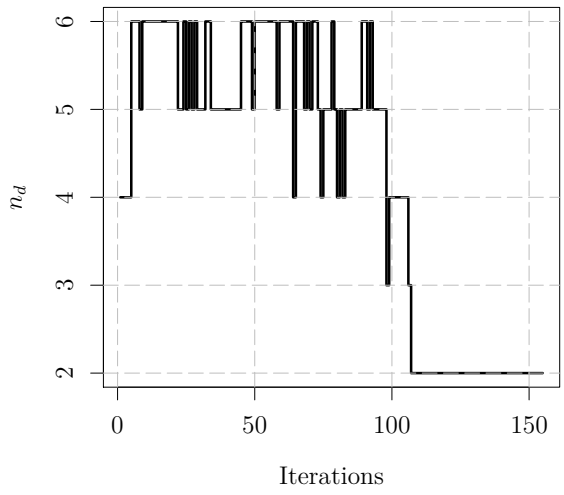


Figure 5.2: A plot of n_d, the number of objects in groups different from their true underlying partitions for the TPEOMA, as the number of iterations (i.e., query pairs) proceed in a single run. Here, $W = 18$, $R = 2$ and $p = 0.9$.

The performance results given above merely compare the TPEOMA to the PEOMA. However, to get an overall perspective of what we have done, a more fair comparison would be to compare the TPEOMA to the original OMA/EOMA. Rather than repeating the results given in the previous papers, in all brevity we state the following: Our results presented here demonstrate that the TPEOMA is about two times faster than the non-transitive versions, and probably about *ninety* times faster than the original OMA.

Table 5.1: Experimental results for the TPEOMA approach done for an ensemble of 100 runs.

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

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

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

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

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

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

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

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

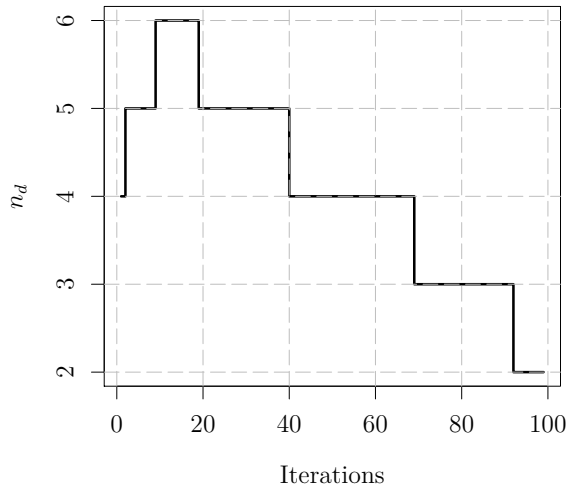


Figure 5.3: A plot of n_d, the number of objects in groups different from their true underlying partitions for the TPEOMA, as the number of iterations (i.e., query pairs) proceed in an ensemble of 100 runs. $W = 18$, $R = 2$ and $p = 0.9$.

5.7 Conclusion

Due to the discrete nature of the partitioning problem and the way the OMA operates, one may think that devising an algorithm which is capable of rewarding objects *without* any explicit feedback from the environment is, if not impossible, very complex. Also, due to the inherent nature of the partitioning problem, increasing the performance of the OMA in near-ideal environments would be challenging, due to the lack of existing queries.

In this research, we have not only demonstrated that such a mechanism exists, but we have proposed an elegant yet simple pursuit-based solution obtaining a linear time complexity overhead, $\mathcal{O}(W)$, compared to the original OMA algorithm. The simulation results have demonstrated that our proposed method improves the PEOMA, the best-reported solution for the EPP, by nearly two times, and this ratio becomes significantly larger for complicated problems and environments. In some problems, our solution is about *ninety* times faster than the solution provided by the OMA.

Further, if we consider unreported problems, the performance gain is even higher!

The core idea of the transitive approach is to predict the behavior of $\mathbb{E}$ and generate *Inferred* queries which would be generated by $\mathbb{E}$ after enough number of iterations with high likelihood. By realizing such an intuition, one can utilize the hidden ties among the objects, and compile virtual reward and penalty responses based on these *Inferred* queries. This is able to significantly increase the convergence ratio in complex partitioning problems. The quantitative analysis of the pursuit matrix in the EPP case, in conjunction with interpreting them qualitatively, reveals the potential of the OMA based solutions in solving the EPP.

In the next chapter, we will deploy the proposed pursuit schemes in a real-life application.

Algorithm 10 TPEOMA

Input:

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

Output:

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

begin
  The initialization of $\{\xi_p\}$, as described in the text.
  for a sequence of $i \leftarrow 1, \cdots, T$ queries **do**
    Read query $\langle A_i, A_j \rangle$
    $\mathcal{Z}[A_i, A_j] \leftarrow \mathcal{Z}[A_i, A_j] + 1$
    $\mathcal{Z}[A_j, A_i] \leftarrow \mathcal{Z}[A_i, A_j]$
    $\mathcal{P}_{i,j} \leftarrow \frac{\mathcal{Z}[A_i, A_j]}{\sum_{k,l=1}^{W} Z[A_k, A_l]}$ ▷ Update stats
    if $i < \kappa$ **then** ▷ Invoke the EOMA
      $SkeletalEOMA(\{\xi_p\}, A_i, A_j)$
    else
      if $\mathcal{P}_{ij} > \tau$ **then** ▷ Valid query
        $SkeletalEOMA(\{\xi_p\}, A_i, A_{ij})$
        for $l \leftarrow \{1, \cdots, W\} \wedge l \neq i, j$ **do**
          if $\mathcal{P}_{il} > \tau_t$ **then**
            $SkeletalEOMA(\{\xi_p\}, A_i, A_l)$
          end if
        end for
      end if
    end if
  end for
  Print out the partitions based on the states $\{\xi_i\}$
end

Algorithm 11 SkeletalEOMA($\{\xi_p\}, A_i, A_l$)

Input:

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

Output:

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

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

Algorithm 12 $ProcessRewardEOMA(\{\xi_p\}, A_i, A_j)$

Input:

- The indices of the states, $\{\xi_p\}$; ξ_i and ξ_j are the states associated with A_i and A_j respectively.
- The query pair $\langle A_i, A_j \rangle$.

Output:

- The next states of the O_i's.

1: **begin**
2: **if** $\xi_i \bmod N \neq 1$ **then** ▷ Move O_i towards the internal state.
3: $\xi_i \leftarrow \xi_i - 1$
4: **end if**
5: **if** $\xi_j \bmod N \neq 1$ **then** ▷ Move O_j towards the internal state.
6: $\xi_j \leftarrow \xi_j - 1$
7: **end if**
8: **return** $\{\xi_p\}$
9: **end**

Algorithm 13 *ProcessPenaltyEOMA*$(\{\xi_p\}, A_i, A_j)$

Input:

- The indices of the states, $\{\xi_p\}$; ξ_i and ξ_j are the states associated with A_i and A_j respectively.
- The query pair $\langle A_i, A_j \rangle$.

Output:

- The next states of the O_i's.

1: **begin**
2: **if** $\xi_i \bmod N \neq 0 \wedge \xi_j \bmod N \neq 0$ **then** ▷ Both are in internal states
3: $\xi_i = \xi_i + 1$
4: $\xi_j = \xi_j + 1$
5: **else if** $\xi_i \bmod N \neq 0$ **then** ▷ O_i is at internal state
6: $\xi_i = \xi_i + 1$
7: $temp = \xi_j$ ▷ Store the state of O_j
8: $l =$ Index of the *unaccessed* object closest to the boundary state of O_i.
9: $\xi_l = temp$
10: $\xi_j = [(\xi_i \boldsymbol{div} N) + 1] \times N$
11: **else if** $\xi_j \bmod N \neq 0$ **then** ▷ O_j is at internal state
12: $\xi_j = \xi_j + 1$
13: $temp = \xi_i$ ▷ Store the state of O_i
14: $l =$ Index of the *unaccessed* object closest to the boundary state of O_j.
15: $\xi_l = temp$
16: **else** ▷ Both are in boundary states
17: $temp = \xi_i$ ▷ Store the state of O_i
18: $\xi_i = \xi_j$ ▷ Move O_i to the same group as O_j
19: $l =$ index of an *unaccessed* object in group of O_j closest to the boundary
20: $\xi_l = temp$ ▷ Move O_l to the old state of O_i
21: **end if**
22: **return** $\{\xi_p\}$
23: **end**

"If the blind lead the blind,
Both will fall into a pit."

The Gospel of Matthew

6

Application: *Noisy* Sensor Networks

6.1 Introduction

In this chapter, we will demonstrate the use of partitioning and OMA-based methods in the field of Noisy Sensor Networks (NSNs).

The adoption of Sensors Networks (SNs) across the industry have led to the development of inexpensive and more effective hardware. The increase in the practical applications of SNs has also introduced a significant number of open problems that need to be investigated. In this regard, we consider the problem of identifying defective sensors in a NSN. Our aim is to incorporate OMA-based approaches, to successfully recognize the sensors that have significantly deviated measurements from the expected values of the "true" state of the Environment. The reported solutions to this problem have, in one way or the other, resorted to using solutions to the so-called "Outlier Detection" problem. The outliers can be due to the noise in the settings in

which the sensor is placed, or due to any other error-causing events, thus affecting their respective reliabilities. Since the operation of SNs have limitations, and since they must meet certain criteria, the traditional "Outlier Detection" methods cannot be applied directly to resolve these issues. Hence, in contrast to the reported solutions, we propose the above-mentioned OMA-based schemes.

As we have seen from the previous chapters, OMA-based solutions are capable of learning the optimal partitioning when operating in unknown stochastic Environments, and additionally, have rapid and accurate convergence with a low computational cost. Unlike most reported approaches in literature, our scheme operates in a Semi-Supervised manner and gradually partitions the sensors. Furthermore, if the quality of the readings and/or the trustworthiness of the sensors change, our method is capable of tracking such changes robustly over time.

6.2 The Field of Noisy Sensor Networks

Within the last decade, interest and research in the area of NSNs has dramatically increased. Indeed, stating that it can be considered to be among the most "researched areas" would not be an overstatement. A comprehensive survey[1] of the theory and the applications of SNs and the methodologies used in utilizing them can be found in [1, 4, 97, 98, 131, 132]

The topology of a SN is usually a distributed network composed of many sensor nodes which collect fine-grained measurements of the physical world. This could include sink nodes which themselves gather pieces of relevant information provided by the sensor nodes themselves.

The sensors in and of themselves are mostly utilized in acquiring data from physical domains and/or in monitoring applications. Generally speaking, the raw sensor data is collected from the nodes and is processed for information extraction and knowledge acquisition. In addition to measuring the phenomena of interest in the

[1]The reader will observe that this thesis is not primarily concerned with Sensor Networks, Wireless Sensor Networks, nor their noisy versions. A survey of the latter fields would thus be far too extensive, and irrelevant in the content of this chapter. Thus, our survey is, necessarily, brief.

domain, the sensors may also be integrated with processors and wireless communication capabilities. SNs can be utilized in a wide spectrum of applications. These applications can extend from personal and business domains to industrial, military or environmental areas [33, 97, 130, 131]. According to the authors of [97], the advantages of wireless SNs over traditional networking solutions can include "*lower costs, scalability, reliability, accuracy, flexibility, and ease of deployment*".

As mentioned above, modern SNs rely on a model of decentralized wireless communications between the autonomous sensor nodes. Such a decentralized network architecture can be beneficial in the case of resolving the unreliable measurements. It also may contribute to a better scalability of the network itself [97, 139].

Recent advancements in technology have led to faster communication and cheaper hardware, thus making SNs more accessible and desirable. Some of the advantages of SNs can be gleaned in the following pieces of literature [1, 4, 97, 98, 131, 132], and include:

- Being high performance hardware, they can provide a significant amount of information about the environment.
- Sensor hardware is usually inexpensive requiring minimal configuration and quick deployment.
- SNs are generally failure resistant due to their decentralized structure, and this leads to dynamic and adaptive routing mechanisms.
- SNs permit the rapid flow of information.

Although SNs are flexible and are powerful tools, there are challenges in their deployment, including:

- Topological inconsistency where the sensors are not located properly. This can occur due to the fact that they are mobile or due to the changes in the environment. In the worst case, the sensors may not be useful anymore.

- Topological resolution in fast-changing environments or when the sensors are too far from each other. In the latter scenarios, they may not communicate fast enough, which leads to inaccurate measurements of the environment.
- High adaptability and routing mechanisms are not always easy to implement and may require expensive hardware.
- Difficulty in operation due to working within a completely distributed manner.
- Reliability of sensors in providing accurate information about the environment may be insignificant.
- There could be problems with obtaining the optimal spatial distributions. In order to keep the costs down, and the readings within acceptable bounds, it may be necessary to design an optimal lattice of nodes so as to maximize the deployment efficiency while providing a full sensing coverage.

As stated above, the data measurements collected by a SN is usually prone to some degree of uncertainty. The unreliability in SNs can originate from internal or external conditions as listed below:

- Environment-related sources: In this case, the existence of noise, extreme conditions and fast dynamics can lead to missing or duplicate values, or to inconsistent measurements.
- Sensor-related sources: In these cases, operational limitations such as power resources, memory, computational constraints and bandwidth can be the cause of erroneous measurements.

As a result of the above, the quality of raw data can be drastically influenced by these factors, rendering the data measurements to be unreliable [110, 139]. Therefore, it is necessary to obtain accurate and reliable measurements, or to recover them accurately from their noisy values, before any decision-making process is invoked. For a further comprehensive overview of the research field in WSNs, we refer the reader to [139].

6.2.1 Chapter Organization

In Section 6.3, we will first describe the fundamentals of the outlier detection problem in NSNs. This will help us to identify the important criteria associated with the classification of outlier detection techniques for NSNs. Thereafter, we will provide a brief technique-based taxonomy to categorize existing techniques developed for NSNs. We will also briefly address the key characteristics, and provide a brief description of the current outlier detection techniques in Section 6.4. In Section 6.5 we will present the details of the NSN partitioning problem. Section 6.6 includes an OMA-based solution to resolve the NSN problem. Section 6.7 summarizes the simulation results and presents a discussion about the performance of our OMA-based solution. Section 6.8 concludes the chapter.

6.3 Fundamentals of Outlier Detection in SNs

In this section, we describe the fundamentals of outlier detection in SNs, including the definitions of outliers, various causes of outliers and their respective *solvability* and *redundancy* issues.

Definition 6.1. "An outlier is an observation, which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [34] or it appears to be inconsistent with the "accepted" set of observations [16].

In the field of SNs, we define outliers as "those measurements that significantly deviate from the normal pattern of sensed data" [20].

Since the nodes in a SN are monitoring physical events, we expect to observe a continuous stream of patterns. Unfortunately, the data gathered by a SN always includes uncertainty, as explained earlier, which is caused by noise and errors. By identifying the causes of anomalies leading to the errors and the corresponding noise, we can either eliminate erroneous readings or correct the noisy observations without a significant impact on the quality of data for the purpose of further data analysis. The potential sources of outliers can be divided into three categories:

- Noise or errors in the sensors: In this case, the physical properties such as the network's communications capabilities, energy, memory and processing power, can affect the performance of the sensors, which in turn, can lead to errors or noise in the observations. Thus, it is desirable to minimize the individual sensor's energy consumption, memory usage and the processing performed, while maximizing/maintaining its battery life and communication capabilities. In some cases, these criteria are contradictory.

- Noise sources or physical events in the environment: In such scenarios, the sensors in the field can be affected by the changes or events in the domain itself. The sensors may become displaced or damaged due to the dynamics in the field. Thus, one should take appropriate measures and even embed extra sensors in order to successfully diagnose these issues.

- External manipulation of sensors: A remarkable example for this scenario involves malicious attacks, which can include, but which are not limited to, manipulating the sensors' software/hardware either remotely or in-place. This scenario requires more advanced diagnostic capabilities embedded in the sensors, and is outside the scope of this study.

6.4 Problem Statement

The majority of the previously-done research has focused on incorporating the information provided by the sensors either under known conditions or by invoking estimated confidence levels. In this section we provide some background information required for the rest of the chapter.

6.4.1 Legacy Methods

Contemporary paradigms describe observations as being random or stochastic processes, and employ probabilistic approaches, such as Bayesian principles, to integrate the noisy observations. Other methods utilize subjective logic [44], evidential reasoning such as Dempster–Shafer Inference theory [17], the weighted average of the

estimated variance of measurements provided by each sensor [25], and somehow even utilize the expert's advice to predict the values of the expected reading [59].

The authors of [62] pioneered the field by proposing the concept of a fault-tolerant averaging algorithm in distributed systems. Majority Voting (MV) algorithms have also been employed to fuse the information gathered from SNs to obtain a single, highly reliable abstract sensor, even when "the information provided by some of the sensors are considered as outliers". Most of the methods in this category of research rely on the Ground Truth to evaluate the accuracy of the sensor. In the MV paradigm the observation obtained from the sensors are compared against the Ground Truth in the training data set[2]. The authors in [41] considered a case in which the readings of the sensors involved in a single observation instance, are divided into two groups, i.e., those that agree with the Ground Truth, and those who are in disagreement with it. The algorithm then determined the *winning* group and increased the reliability factor for this group of sensors (which can be considered as a "reward"), and simultaneously decreased the reliability factor for the losing group (which can be considered as a "penalty").

6.4.2 State-of-the-Art Schemes: Majority Methods

Within the past decade, with the increase in the deployment of SN technology in the industry, obtaining a robust and unattended measurement has been recognized as a central topic of research.

A MV is a binary decision mechanism which chooses the alternative that sides with the majority, i.e., with more than half the votes. This paradigm is based on Boland's idea that "the majority group is always better at selecting superior alternatives than any single individual member" [18]. Although this paradigm is simple and elegant, it has some limitations for it to be utilized in real-world applications. In the problem of SNs, it is necessary that each sensor reports the correct value with the probability $p > 0.5$ through all of the experiments[3]. Considering the above-mentioned criteria, we consider two categories of sensors namely, the sensors which report the "true"

[2]Our OMA-based scheme works analogously.

[3]Typically, in the literature, all of the "reliable" sensors share the same probability value, p.

value of the domain and the category which are opposed to this true value. In other words, if we consider the first group of sensors to be "reliable", i.e., which tend to report the "true" state of the domain, and the second group consists of those who report the state of the domain incorrectly.

6.5 Problem Statement

We first consider a group of N sensors, $S = \{s_1, s_2, \cdots, s_N\}$ configured as in Figure 6.1. We assume that the true state of the domain at any time instant t, is represented by a binary value $T(t) \in \{0, 1\}$. Obviously, the value of $T(t)$ is unknown, and can only be inferred through the measurements obtained from the sensors. We denote the output of the sensor s_i, to be x_i. Let π represent the probability of the Ground Truth being $T \in \{0, 1\}$. With this as the basis, we can consider two cases, each consisting of two possibilities:

1. $x_i = T$. In this case, the sensor has accurately reported the Ground Truth to be either 0 or 1.

2. $x_i \neq T$. Here, the sensor has reported a value incompatible with the Ground Truth (again 0 or 1).

In order to simplify the discussion here, we assume that the probability for each sensor to report a faulty reading is symmetric for both the cases, i.e., $T = 0$ and $T = 1$ cases. Thus

$$\Pr = (x_i = 0|T = 1) = \Pr(x_i = 1|T = 0). \tag{6.1}$$

Let q_i be the probability of sensor s_i being faulty, referred to as the Fault Probability (FP), where

$$q_i = \Pr(x_i = 0|T = 1) = \Pr(x_i = 1|T = 0). \tag{6.2}$$

Similarly, we can define the probability of the sensor s_i to be operating correctly, referred to as the Correctness Probability (CP), as $p_i = 1 - q_i$. The total probability of $\Pr(x_i = T)$ is thus equal to p_i since:

$$\begin{aligned}\Pr(x_i = T) &= \Pr(T = 0)\Pr(x_i = 0|T = 0) + \Pr(T = 1)\Pr(x_i = 1|T = 1)\\ &= \pi p_i + (1 - \pi)p_i = p_i.\end{aligned}$$

We can rewrite the above as $p_i = \Pr(I\{x_i = T\} = 1)$, where $I\{.\}$ is the *Indicator* function. As discussed previously, we consider a sensor unreliable if its relevant FP, $q_i \geq 0.5$ or its CP is $p_i \leq 0.5$. On the other hand, a sensor is reliable if its CP, $p_i > 0.5$. It can be easily concluded that a reliable sensor will most likely report 0 when the Ground Truth is 0, and a faulty sensor will report 1. Since the unreliable sensors will have impact on the quality of the sensed data, it is desirable to recognize these sensors, which can be viewed as a task of partitioning the sensors into reliable and non-reliable groups. To facilitate the analysis, we assume[4] that $p_i \in \{p_R, p_U\}$, where $p_R > 0.5$ and $p_U = 1 - p_R$. Following the above-mentioned formulation, a sensor is reliable if $p_i = p_U$. These two quantities are unknown, except that we know $p_R > p_U$. We represent the set of reliable sensors as $\mathcal{S}_R = \{s_i|p_i = p_R\}$ and the set of non-reliable sensors as $\mathcal{S}_U = \{s_i|p_i = p_U\}$. In our formulation[5], we work with the setting that the cardinalities of these two sets are equal, i.e., $|S_R| = |S_U|$.

The method that we propose can be accurately referred to as a *Semi*-Supervised scheme. This is because we first utilize the readings of the Ground Truth to obtain an initial placement of the sensors. This phase is totally Supervised. Once we can assert, with a reasonable certainty, that the sensors are in their right groupings, we can work without invoking the Ground Truth, that served as the Oracle. We then can enter the Unsupervised phase, where the reading of the sensor is compared to the readings of the majority of the sensors, and thus every sensor can be aligned based on such a consensus.

[4]This assumption is realistic since p_R can be set to be the minimum vase of p_i for the reliable scenarios, and p_U can be set to be the maximum value of p_i for the unreliable sensors.

[5]The setting where $|S_U| \neq |S_R|$ can be resolved if the cardinality of the sets are known. Otherwise, i.e., if the cardinalities are not known, the OMA-based solution remains open.

6.6 OMA-based Scheme

In this section, we propose an OMA-based solution to the problem of partitioning the NSs. We propose to use an EOMA to address the corresponding EPP in a semi-supervised manner. In this scenario, the sensors are initially distributed randomly in to two partitions, which are *Reliable*, denoted by S_R, and *Unreliable* specified by S_U. At every iteration, all of the sensors sense the domain, and report their values. Thereafter, a sensor is selected at random and its reading is compared to the Ground Truth. If the sensor's output agrees with the current true state of the domain, it will be rewarded or penalized based on the partition in which it resides. For example, if a sensor is in the *Unreliable* partition but it agrees with the Ground Truth, it must be penalized. The same applies to the case when the sensor reports the wrong value but it is located in the *Reliable*-sensors partition.

Since there is no Environment such as the one assumed in the previous chapters, we encounter a dilemma namely, that of understanding how we can infer which group a particular sensor should be migrated. Observe that this would have been a trivial problem if the sensors were paired by an Environment, and then presented to an OMA-based algorithm. The dilemma is further complicated by the fact that all the W sensors can report their measurements, and the learning machine would thus be unable to make any meaningful decisions about the pairings because there would be $\binom{W}{2}$ such pairs. Further, the resolution of the dilemma must be done on-line.

The strategy by which we circumvent this problem is by operating on the same level field as the prior art, namely, by assuming the existence of an additional sensor, which provides the Ground Truth. We shall refer to this sensor as the Oracle. Thus, in effect, we are dealing with $W + 1$ sensors (where W is even), from among which one is noiseless and is the Oracle, and the remaining must be equi-partitioned into the sets S_R and S_U, both of cardinality of $\frac{W}{2}$. Unlike the state-of-the-art however, we will not require that the Oracle provides us with the Ground Truth endlessly. Rather, in the initial "training" phase of the machine, the Oracle provides the Ground Truth to enable us to achieve the partitioning. This constitutes a *Supervised* mode of learning. After this transient phase, we need not depend on the Oracle anymore. The

W sensors will already be partitioned into their sets S_R and S_U, and a simple MV scheme will suffice to the further facilitate the convergence. The latter phase models an *Unsupervised* mode of learning. Thus, between the two, we can characterize the scheme as being "semi-supervised".

Let us now consider the transient *Supervised* mode. The Oracle providing the Ground Truth is made to reside in the innermost state of the OMA-based machine associate with S_R. At every time instant, we randomly pick a sensor and check if its reading is identical to the Oracle's. Notice that if it is a *Reliable* sensor, this occurs with the probability p_R. However, an *Unreliable* sensor could also align itself with the Oracle with the probability $1 - p_U$. It is here that the stochastic learning capabilities of the OMA-based machine become crucial. If the value of the randomly-chosen sensor aligns with the Oracle, and it is in the set S_R, the sensor will be *rewarded* and will be moved deeper into its partition. However, if it is located in the set associated with S_U, it is residing in the wrong partition, and will be *penalized*. Here, the reward and penalty functions are identical to what the OMA dictates. Analogously, if the randomly chosen sensor does not align itself with the Oracle, it is stochastically reckoned to be faulty. This can happen to an *Unreliable* sensor, with probability p_U, and to a *Reliable* sensor with probability $1-p_R$. Again, if this sensor is in the same set as the Oracle, it is *penalized* by moving away from the Oracle towards the boundary state. At the boundary state, it is moved into the boundary state of the *Unreliable* sensors in an OMA-like fashion. On the other hand, if it is found in the opposite class of the Oracle, the particular sensor is *rewarded* and moves towards its innermost state. By operating in such a manner, we are capable of providing OMA-like reward and penalty transitions without requiring an unending pair of queries – as was needed in the earlier chapters. Rather, all that we require is the reading of a single randomly chosen sensor and the existence of the Oracle for the transient phase.

After a reasonable number of readings, the Oracle is not required and we can choose a random sensor and compare its reading with the MV. Once the sensors have converged into their two goups, the decision of the majority can be seen to be the negation of the decision of the Oracle[6].

[6]This is ironic but it is, nevertheless, true.

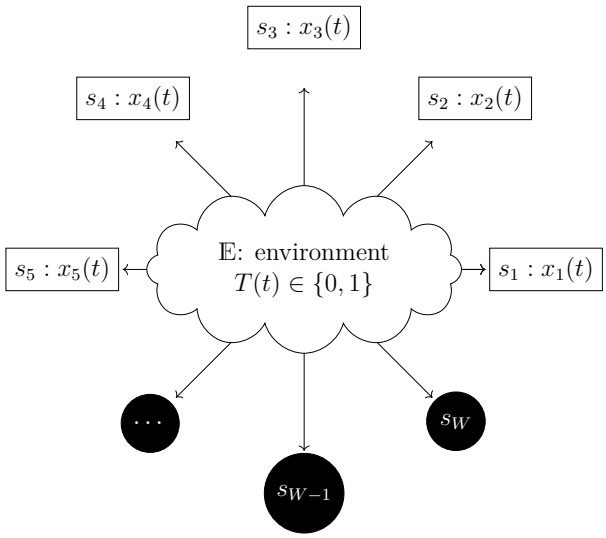


Figure 6.1: A figure describing the network of sensors interacting with the domain in which the measurements are being taken.

The formal algorithms for the overall scheme is given in Algorithm 14, and the reward and penalty scenarios are provided in Algorithms 15 and 16 respectively.

In the next section, we discuss the simulation results.

6.7 Simulation Results and Convergence

The simulation results obtained by introducing all of the three above-mentioned modifications are given in Table 6.1. In this table, we have reported the results obtained for various settings, with probabilities $p_R = 1.0, \cdots, 0.7$, where $p_R = 1.0$ represents the optimal ideal sensor. On the other hand, for the case where $p_R = 0.7$, we encounter a difficult-to-learn scenario. All the simulations reported were done on an ensemble of 100 experiments to guarantee statistically stable results.

While the EOMA-based algorithm is migrating sensors between the S_R and S_U groups, there will be an iteration index at which juncture all the objects reside in their current, accurate partitioning. The number of iterations that have taken place up to this point, which relates to the first time when the Sensor-EOMA has divided the correct goups of the sensors, is recorded. Additionally, we have also recorded

Algorithm 14 Sensor-EOMA

Input:

- A set of W sensors, $\mathcal{S} \leftarrow \{s_1, s_2, \cdots, s_W\}$.
- The number of states W per action.
- A group of W sensed outputs, $\langle x_1, x_2, \cdots, x_W \rangle$.
- A stream of the domains status provided by the oracle.
- r to control the convergence of the algorithm.

Output:

- A periodic clustering of the sensors into R partitions.
- ξ_i is the state of the sensor s_i, and i is an integer ranging from $1 \cdots N$.
- The set $\mathcal{T} \leftarrow \{t_1, t_2, \cdots, t_W\}$, which represents the true state of the domain provided by the oracle.

1: **begin**
2: $r \leftarrow 1$
3: Initialization of $\{\xi_p\}$ ▷ Described in the text
4: **while** $r \neq 0$ **do**
5: read($\{x_1, x_2, \cdots, x_W\}$) ▷ Read sensor values
6: $k \leftarrow U(1, N)$ ▷ Choose a random index
7: **if** $\{(\xi_k = t_i) \wedge (\xi_k \in S_R)\} \vee \{(\xi_k \neq t_i) \wedge (\xi_k \in S_U)\}$ **then**
8: $SensorReward(x_k)$ ▷ Reward case
9: **else**
10: $SensorPenalty(x_k)$ ▷ Penalty case
11: **end if**
12: **if** $\forall k : \xi_k \mod N \leq 2$ **then** ▷ In last two innermost states
13: $r = 0$ ▷ Converged: Exit while loop
14: **end if**
15: **end while**
16: **end**

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 Sensor-EOMA to achieve the characterization. To be more specific, consider the case given in Table 6.1, where it took the Sensor-EOMA 192 iterations to settle into the final

Algorithm 15 SensorReward(x_i)

Input:

- Current index of the sensor.
- N, the number of states in each partition.
- The set of $\{\xi_1, \xi_2, \cdots, \xi_W\}$ to represent the state of the sensors in each partition.

Output:

- Updated sensor state.

```
1: begin
2:     if State of ξ_i mod N ≠ 1 then                ▷ Not the most internal state
3:         ξ_i ← ξ_i − 1                             ▷ Decrease the state's value
4:     end if
5: end
```

Algorithm 16 SensorPenalty(x_i)

Input:

- Current index of the sensor.
- N, the number of states in each partition.

Output:

- Updated sensor state.

```
1: begin
2:     if ξ_i mod N > 0 then                         ▷ ξ_i is the most
3:         ξ_i ← ξ_i + 1                             ▷ Decrease the state's value
4:     else if ξ_i mod N = 0 then
5:         i ← ξ_i mod N                             ▷ Store state of x_i
6:         I ← {∀j : ξ_j mod N ≠ i}                  ▷ All indeces from the opposite partition
7:         if length(I) > 0 then                     ▷ Not an empty set
8:             I' ← sort(I, decreasing)              ▷ Sort decreasingly
9:             swap(ξ_i, ξ_I'[1])                    ▷ Swap it with the first element
10:        end if
11:    end if
12: end
```

configuration ($p_r = 0.8$). Here, it took the Sensor-EOMA only 22 iterations to first reach the correct classification. In such a case, the algorithm would report the pair $\langle 22, 192 \rangle$ as the Sensor-EOMA's convergence results.

From Table 6.1, we observe that as the value of p_R increases, the queries are more informative, and the convergence occurs at a faster rate. As before, the complexity of the classification problem has two criteria. Due to the constraints that we imposed in the model, we have $\frac{W}{2}$ objects in each group, and so, as the number of sensors increases, the problem becomes increasingly complex to solve.

By way of example, consider the entries in Table 6.1 for the case when p_R and $W = 30$, and when the probability of a *Reliable* sensor reporting in accordance with the Ground Truth was $p_R = 0.8$. It took the Sensor-EOMA, on average, $1,129$ iterations to converge to the correct partitioning, while it took only 240 readings for the sensors to fall into their accurate partitions.

Table 6.1: Simulation Results for the EOMA. The reported results in this table are the average ensemble of 100 runs.

W	$p_R = 1$	$p_R = 0.9$	$p_R = 0.8$	$p_R = 0.7$
3,3	7,74	8,100	11,141	12,143
4,4	11,104	13,139	22,192	39,389
6,6	22,166	29,221	53,348	107,754
8,8	32,229	49,309	88,494	169,1365
10,10	48,298	65,413	118,640	251,1933
15,15	84,470	124,682	240,1129	526,9191
20,20	123,647	189,911	341,1832	657,45739

p_R: Represents the probability of a sensor being reliable.
W: Represents the total number of sensors divided in to two groups as discussed in the text.

As one can easily observe, both the Oracle and the Sensor-EOMA can easily operate in extremely complex (highly noisy) domains, and solve difficult problems rapidly. Before we conclude this section, we would like to make the following remarks:

- As can be seen from the table, the problem of characterizing sensors becomes

increasingly easy, as the value of p_R increases (or p_U decreases);

- In the above table, we have deliberately chosen to use the Oracle even after all the sensors are in their accurate partitions. This is a luxurious choice. Rather, after we can infer that the Oracle is not needed anymore, we could resort to an MV strategy to force the sensors to the innermost states of each group;

- Throughout this entire chapter, the reader will observe, that we have merely used the EOMA to achieve the partitioning. The novel phenomena of utilizing the pursuit and transitivity properties have not been invoked. We believe that, if the PEOMA and/or the TPEOMA were utilized instead of the EOMA, the convergence of the sensors would have been much more rapid. This is currently and unresolved problem primarily because the pursuit matrix requires *pairs* of query objects whose joint access probabilities are estimated. This is not realistic in the current sensor partitioning model as we only examine one sensor at a time;

- Finally, the reader will observe that in Algorithm 14, we have deliberately chosen to examine the reading of a single sensor chosen at random (line 6 in Algorithm 14). Obviously, if the readings of multiple sensors were taken, the algorithm could be forced to converge even faster.

6.8 Conclusion

In this chapter, we have considered the problem of partitioning noisy sensors, which are either *Reliable* or *Unreliable*. If they are *Reliable*, they report the precise value with the probability of $p_R > 0.5$. Similarly, the *Unreliable* sensors report the true value with the probability of $p_U \geq 0.5$. By considering the reading of only a single sensor at a time, and the existence of an Oracle reporting Ground Truth, we have been able to design an OMA-like mechanism to accurately equi-partition the sensors.

The learning machine in this case has been the EOMA, described in Chapter 4. We have not considered the use of the PEOMA or the TPEOMA in achieving the

partitioning. This problem is currently unresolved, because we have not assumed the existence of an Environment which provides intelligent information about *pairs* of sensors.

Finally, although we have required the cardinalities of S_R and S_U to be equal, with a little imagination, one can extend our present scheme to the case where the *sizes* of S_R and S_U are unkown. The problem of achieving the partitioning when the sizes of S_U and S_R are unkown is far more complex and is currently unsolved.

"The woods are lovely, dark, and deep,
But I have promises to keep,
And miles to go before I sleep,
And miles to go before I sleep."

Robert Frost

7

Summary, Conclusions and Future Work

This chapter provides a summary of the work we have done, submits our conclusions, and gives a few ideas for future research.

7.1 Summary and Conclusions

7.1.1 Summary of the Chapters

In Chapter 2, we surveyed the fields of Learning Automata (LA) and Machine Learning (ML). We first explored the concept of "Learning" from a psychological perspective. We also mentioned the influence of psychology on the area of ML. We then visited the necessary aspects of ML and statistics such as *model selection* and *parameter estimation*. We also surveyed the field of LA, and demonstrated how it is strongly related to concepts in the theory of "Learning". In this chapter, we showed

that the general model of LA can be characterized and defined both quantitatively and qualitatively, and demonstrated how the learning characteristics of an LA can be measured. We reviewed two major classes of LA, i.e., that of FSSA and the VSSA, and thus covered the details of each of these families, in some detail.

We also discussed the convergence and the performance of LA in addition to their properties in multi-teacher and non-stationary random environments. We discussed how Maximum Likelihood Estimation (MLE) methods could be used in estimating the reward probabilities so as to enhance VSSA – an endeavor which led to the families of Estimator and Pursuit LA.

Chapter 3 focused on the OMA. In this chapter, we introduced the OPP, and studied a special case where the number of objects in each group are equal, which is referred to as the EPP. We further reviewed the previously-proposed methods for the OPP and the EPP.

With regard to the EPP, we discussed, in detail, the OMA, which is an LA-based approach to resolve the EPP. This solution is regarded as a benchmark.

Moving now on to the Environment's behavior in generating the query stream, we proposed two models for simulating both noisy and noiseless Environments. We argued that the so-called "divergent" queries can drastically slow down the convergence rate of the OMA (and in general, any partitioning algorithm). 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 POMA.

We also discussed the details of the Pursuit paradigm, such as obtaining a fair estimate for the POMA's threshold value, τ. Also, with regard to demonstrating the strength of the Pursuit techniques, we presented the experimental results obtained, on benchmark environments, that compared the POMA and the OMA. The results show that the POMA out-performed 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 convergent query was $p = 0.7$, the POMA performed nearly two times better than the OMA. 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.

The results reported in Chapter 3 are published in [106], and its extended journal version has been published in [103].

Chapter 4 reviewed the Enhanced version of the OMA (EOMA), which addresses the deadlock issue in the presence of a large number of actions and a high level of noise, and the task of optimizing it by utilizing the Pursuit concept. This led to the PEOMA. We compared this proposed method to other benchmark schemes.

The PEOMA utilized the exact same Pursuit principles used in the POMA in Chapter 3. We argued that the so-called "deadlock" situation in addition to the "divergent" queries, would be able to drastically slow down or even, in some rare cases, stop the convergence of the OMA. Consequently, we considered how the deadlock problem could be resolved. We then combined it with the Pursuit concept that was used as an effective accept/reject policy, which in turn, led to a significant gain in the performance of the original OMA. This was also able to take care of the scenarios when we had a large number of groups and a very noisy Environment.

The experimental results compared the PEOMA with the EOMA, and we were able to show how the PEOMA out-performed the EOMA and the OMA. By studying a spectrum of problems, which extend from easy-to-learn to difficult Environments, and for simple or complex partitioning problems, we realized that the PEOMA's performance can be up to more than *two* times better than the EOMA. But if we compare the results with the original OMA, the immense performance gain leads to about *forty* times less number of iterations for a complete convergence – which is by no means insignificant. It is fascinating to note that the reduction in the number of iterations required by the PEOMA can again be seen to be a consequence of a Pursuit-like filtering phase in all problem domains.

In summary, the introduced modifications in Chapter 4, enhanced the OMA's performance without any significant computational overhead in all the examined Environments, even when the partitioning problem was inherently challenging. The results mentioned in Chapter 4 are published in [104], and its extended journal version has been revised after a peer review and is currently being considered for publication

[105].

In Chapter 5, we proposed an elegant yet simple Pursuit-based solution which obtains a linear time complexity overhead, $\mathcal{O}(W)$, compared to the original OMA. Our proposed method was capable of rewarding objects *without* any explicit feedback from the Environment. Despite the inherent nature of the partitioning problem, we were able to increase the performance of the OMA in near-ideal environments which is a challenging task, due to the lack of existing queries.

The simulation results have demonstrated that our proposed method improves the PEOMA, the best-reported solution for the EPP, by nearly two times, and this ratio becomes significantly larger for complicated problems and environments. In some problems, our solution is about *ninety* times faster than the solution provided by the original OMA. Further, if we consider certain unreported scenarios, the performance gain is even higher!

The core idea of the transitive approach is to predict the behavior of Environment, $\mathbb{E}$, and to thus generate *Inferred* queries which could have been generated, after enough number of iterations have been encountered. By realizing such an intuition, one can utilize the hidden ties among the objects, and compile "virtual" reward and penalty responses based on these *Inferred* queries. This is able to significantly increase the convergence ratio in complex partitioning problems. The quantitative analysis of the Pursuit matrix in the EPP, in conjunction with interpreting them qualitatively, reveals the potential of the OMA-based solutions in solving the EPP. The results reported in Chapter 5 have been accepted for publication [107].

7.1.2 Conclusions of the Thesis

The conclusions of the thesis can be listed as follows:

- The OMA, as reported earlier in the existing literature, can be used as a tool to resolve the EPP;
- There is a vast amount of information that has not been utilized earlier in the existing literature, and this is resident in the so-called Pursuit matrix;

- The information contained in the Pursuit matrix can be estimated and used to increase the convergence speed of the original OMA;
- This has been achieved in this research endeavor by designing the so-called PEOMA;
- The information in the Pursuit matrix can also be used to make the Enhanced OMA faster. The latter, which was proposed to mitigate the deadlock scenario of the OMA, can also be improved by utilizing this information;
- The EPP possesses the inherent property of transitivity. We can, thus, generate *Inferred* queries using the information in the same Pursuit matrix to further catalyze the performance of the corresponding OMA-based mechanism;
- OMA-based mechanisms can be designed and implemented to resolve the outlier detection problem for Noisy Sensor Networks.

7.2 Future Work

We now briefly discuss the unexplored potential research areas which can be considered as future avenues for investigation. From the algorithmic point of view, the partitioning problem manifests itself in different configurations and complexities. Some of the relevant open cases are when:

- The number of objects to be partitioned and the number of partitions available are unknown;
- The number of objects *within* each of the partitions is unknown.

The restricted version for the above has been addressed, to some extent, in the literature. The general case, however, has not been considered here, and is actually reckoned to be open. Indeed, the focus of almost all of the previously-reported research concentrated on a special case of the OPP, i.e., the EPP.

We now consider two other possible configurations for the partitioning problem, namely:

- The Unbalanced Partitioning Problem (UPP), when there is no constraint on the number of objects in the groups, but where the number of objects in the various groups is specified;
- The Proportional Partitioning Problem (PPP), in which the number of objects in the various groups have common divisors.

In the following we will expand on each of the corresponding problems, and state the future goals of possible research in these areas.

7.2.1 Unbalanced Partitioning Problem(UPP)

As mentioned, all solutions that utilized the OMA have focused on solving the EPP in which all the objects to be partitioned are equally distributed among the predefined number of classes. Removing this constraint leads to two new categories of problems. We first briefly define the UPP.

In the UPP, there are no constraints on the number objects that should occur in the various groups. Thus, the problem takes the general format of classifying W objects into R classes, with each group containing at least two objects:

$$\sum_1^R K_i \quad = W; \gcd\left(K_1, K_2, \cdots, K_R\right) \quad = 1, \tag{7.1}$$

where K_i represents the number of objects in group i, and where it is not required that $\forall i \in \{1, 2, \cdots, R\},\ K_i = \frac{W}{R}$. As an example, if we have 12 objects and 4 groups, we can partition them into groups with cardinalities $\{2, 3, 5, 2\}$. We do, however, provide as an input to the algorithm, the information that the Environment generates the objects as from sets with these underlying cardinalities, which are unknown, though, to the LA.

In future research, we intend to study this problem for all of the original, enhanced, and Pursuit versions of the OMA-like schemes. Solving such UPP problems will require introducing new algorithmic modifications, and manipulating the structure of the OMA so as to make the new LA capable of resolving the facet due to the added complexity.

We had earlier mentioned that a statistical and algebraic analyses of the relations among objects can yield a large source of information with respect to the partitioning problem. The goals of the future research would be to relate these structural properties (i.e., the statistical or algebraic) and the partitions that they yield. As an example, since the Pursuit matrix represents the adjacency matrix from a statistical point of view, it can be easily utilized to represent to extract not only time-driven measures, but also the corresponding spectral properties of the relations of this "graph", i.e., its characteristic polynomial, eigenvalues and eigenvectors. All of these issues are currently unsolved.

7.2.2 Proportional Partitioning Problem (PPP)

In the PPP, we further remove the constraint given by Equation (7.1). Here, the only constraint that is enforced is that the number objects in every group should have a common divisor. Thus, the problem we attempt to resolve here is to group W objects into R classes such that:

$$\sum_1^R K_i \quad = W; \gcd\left(K_1, K_2, \cdots, K_R\right) \quad > 1, \tag{7.2}$$

where, as before, K_i is the number of objects in group i.

Although the UPP can be considered as a "superset" problem of the PPP, what distinguishes it from the former is that we could now introduce new algorithmic aspects. We also intend to study the differences between the respective algorithms, i.e., both performance-wise and precision-wise. Since, in the PPP, we can take advantage of the proportional relationship between the groups, it is likely that its solution can be obtained more easily.

Again, we expect to study this problem for all of the original, enhanced and Pursuit versions of OMA-like LA. This too would involve introducing algorithmic changes into the LA and/or manipulating the structure of the machine itself.

7.2.3 Investigating Other Possible Real-world Physical Applications

As explained in the core of the thesis, our study in partitioning is not merely theoretical. We applied the proposed methods to the Noisy Sensor Networks (NSNs) outlier detection problem. Indeed, as suggested in Chapter 3, the OMA has been used to solve many other real-world problems. Since the Pursuit versions of the OMA and their transitive versions operate in "higher dimensions" by incorporating the knowledge gleaned about the Environment, we hypothesize that they are, truly, suitable candidates for all of the applications in which the OMA has been utilized. As our future research plan, we hope to evaluate the experimental results of incorporating them in these problems, and intend to compare our newly-obtained schemes to the previously-reported OMA-based solutions.

Bibliography

[1] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer communications*, 30(14-15):2826–2841, 2007. doi: https://doi.org/10.1016/j.comcom.2007.05.024.

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

[3] M. Agache and B.J. Oommen. Generalized tse: a new generalized estimator based learning automaton. In *The IEEE Conference on Cybernetics and Intelligent Systems*, volume 1, pages 245–251. IEEE, 2004.

[4] I. F Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, 2002. doi: https://doi.org/10.1109/MCOM.2002.1024422.

[5] A. Amer and B. J. Oommen. A novel framework for self-organizing lists in environments with locality of reference: Lists-on-lists. *The Computer Journal*, 50(2):186–196, 2007.

[6] F. Annexstein, M. Baumslag, and A. L. Rosenberg. Group action graphs and parallel architectures. *SIAM Journal on Computing*, 19(3):544–569, 1990.

[7] H. Aso and M. Kimura. The structures of automata to adapt to an unknown environment. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(7):494–504, 1976.

[8] H. Aso and M. Kimura. Absolute expediency of learning automata. *Information Sciences*, 17(2):91–112, 1979.

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

[10] 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.

[11] 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.

[12] N. Baba. Learning behaviors of hierarchical structure stochastic automata operating in a general multiteacher environment. *IEEE Transactions on Systems, Man, & Cybernetics*, 1985.

[13] N. Baba and Y. Mogami. A new learning algorithm for the hierarchical structure learning automata operating in the nonstationary s-model random environment. *IEEE transactions on systems, man and cybernetics. Part B, Cybernetics*, 32(6):750–758, 2002.

[14] N. Baba and Y. Mogami. A relative reward-strength algorithm for the hierarchical structure learning automata operating in the general non-stationary multi-teacher environment. *IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics)*, 36(4):781–794, 2006.

[15] N. Baba and Y. Sawaragi. On the learning behavior of stochastic automata under a nonstationary random environment. *IEEE Transactions on Systems Man and Cybernetics*, 2(SMC-5):273–275, 1975.

[16] V. Barnett and T. Lewis. *Outliers in statistical data*. Wiley, 1974.

[17] P. L. Bogler. Shafer-dempster reasoning with applications to multisensor target identification systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(6):968–977, 1987. doi: https://doi.org/10.1109/TSMC.1987.6499307.

[18] P. J. Boland. Majority systems and the condorcet jury theorem. *The Statistician*, pages 181–189, 1989. doi: https://doi.org/10.2307/2348873.

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

[20] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009. doi: https://doi.org/10.1145/1541880.1541882.

[21] B. Chandrasekaran and D. WC. Shen. On expediency and convergence in variable-structure automata. *IEEE Transactions on systems science and cybernetics*, 4(1):52–60, 1968.

[22] Ms Vibhavari Chavan and Rajesh N Phursule. Survey paper on big data. *International Journal of Computer Science and Information Technologies, IJCSIT*, pages 7932–7939, 2014.

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

[24] T. Cover and M. Hellman. The two-armed-bandit problem with time-invariant finite memory. *IEEE Transactions on Information Theory*, 16(2):185–195, 1970.

[25] W. Elmenreich. Fusion of continuous-valued sensor measurements using confidence-weighted averaging. *Journal of Vibration and Control*, 13(9-10):1303–1312, 2007. doi: https://doi.org/10.1177/1077546307077457.

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

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

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

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

[30] M. S. Georgios, I. Papadimitriou, and A. S. Pomportsis. A new class of ε-optimal learning automata. *IEEE Trans. Syst. Man Cybern*, 34(1):246–254, 2004.

[31] M. Hammer and A. Chan. Index selection in a self-adaptive data base management system. In *The International Conference on Management of Data (SIGMOD)*, pages 1–8. ACM, 1976.

[32] M. Hammer and B. Niamir. A heuristic approach to attribute partitioning. In *The International Conference on Management of Data (SIGMOD)*, pages 93–101. ACM, 1979.

[33] P Harrop. Wireless sensor networks and the new internet of things. *Energy Harvest J*, 2012.

[34] D. M. Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

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

[36] G. Horn and B. J. Oommen. A fixed-structure learning automaton solution to the stochastic static mapping problem. In *The 19th IEEE International Parallel and Distributed Processing Symposium*, pages 297b–297b. IEEE, 2005.

[37] G. Horn and B. J. Oommen. Generalised pursuit learning automata for non-stationary environments applied to the stochastic static mapping problem. In

The 11th International Conference on Information Systems Analysis and Synthesis (ISAS) and The 2nd International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA), volume 1, pages 91–97, 2005.

[38] G. Horn and B. J. Oommen. Towards a learning automata solution to the multi-constraint partitioning problem. In *The IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–8. IEEE, 2006.

[39] G. Horn and B. J. Oommen. Towards a learning automata solution to the multi-constraint partitioning problem. In *The IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–8. IEEE, 2006.

[40] G. Horn and B. J. Oommen. Solving multi-constraint assignment problems using learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(1):6–18, 2010.

[41] M. A. Hossain, P. K. Atrey, and A. El-Saddik. Learning multisensor confidence using a reward-and-punishment mechanism. *IEEE Transactions on Instrumentation and Measurement*, 58(5):1525–1534, 2009. doi: `https://doi.org/10.1109/TIM.2009.2014507`.

[42] R. Iraji, M. Manzuri-Shalmani, A. H. Jamalian, and H. Beigy. Ija automaton: Expediency and ϵ-optimality properties. In *The 5th International Conference on Cognitive Informatics*, volume 1, pages 617–622. IEEE, 2006.

[43] A. Jobava. Intelligent Traffic-aware Consolidation of Virtual Machines in a Data Center. Master's thesis, University of Oslo, 2015.

[44] A. Jøsang. Artificial reasoning with subjective logic. In *The Second Australian Workshop on Commonsense Reasoning*, volume 48, page 34. Citeseer, 1997. doi: `https://doi.org/10.1.1.614.5935`.

[45] J. Kabudian, M. R. Meybodi, and M. M. Homayounpour. Applying continuous action reinforcement learning automata (carla) to global training of hidden

markov models. In *International Conference on Information Technology: Coding and Computing (ITCC)*, volume 2, pages 638–642. IEEE, 2004.

[46] N. KANDELAK and TSERTSVA. G. Behavior of certain classes of stochastic automata in random media. *AUTOMATION AND REMOTE CONTROL*, 27(6):1037, 1966.

[47] B. W Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[48] D. E. Koditschek and K. S. Narendra. Fixed structure automata in a multi-teacher environment. *IEEE Transactions on Systems, Man and Cybernetics*, 7(8):616–624, 1977.

[49] V. Krinskii. Asymptotic optimal automation with exponential speed of convergence. *Biofizika*, 9:484, 1964.

[50] V. Y. Krylov. One stochastic automaton which is asymptotically optimal in random medium. *Automation and Remote Control*, 24:1114–1116, 1964.

[51] K. B. Lakshmanan and B. Chandrasekaran. Compound hypothesis testing with finite memory. *Information and Control*, 40(2):223–233, 1979.

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

[53] S. Lakshmivarahan and M. A. L. Thathachar. Bayesian learning and reinforcement schemes for stochastic automata. In *The International Conference on Cybernetics and Society*, volume 134, 1972.

[54] S. Lakshmivarahan and M. A. L. Thathachar. Optimal non-linear reinforcement schemes for stochastic automata. *Information Sciences*, 4(2):121–128, 1972.

[55] S. Lakshmivarahan and M. A. L. Thathachar. Absolutely expedient learning algorithms for stochastic automata. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, SCM-6(3):281–286, 1973.

[56] S. Lakshmivarahan and M. A. L. Thathachar. Absolute expediency of q-and s-model learning algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(SMC-6):222–226, 1976.

[57] S. Lakshmivarahan and M. A. L. Thathachar. Bounds on the convergence probabilities of learning automata. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 6(11):756–763, 1976.

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

[59] Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994. doi: `https://doi.org/10.1006/inco.1994.1009`.

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

[61] Ryan Martin and Omkar Tilak. On ϵ-optimality of the pursuit learning algorithm. *Journal of Applied Probability*, 49(3):795–805, 2012.

[62] Keith Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems (TOCS)*, 8(4):284–304, 1990. doi: `https://doi.org/10.1145/128733.128735`.

[63] G. McMurtry and K. Fu. A variable structure automaton used as a multimodal searching technique. *IEEE Transactions on Automatic Control*, 11(3):379–387, 1966.

[64] 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.

[65] 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.

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

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

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

[69] K. S. Narendra and R Viswanathan. A two-level system of stochastic automata for periodic random environments. In *The IEEE Conference on Decision and Control*, pages 234–239. IEEE, 1971.

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

[71] M. F. Norman. Slow learning. *British Journal of Mathematical and Statistical Psychology*, 21(2):141–159, 1968.

[72] M. F. Norman. A central limit theorem for markov processes that move by small steps. *The Annals of Probability*, pages 1065–1074, 1974.

[73] 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. doi: `https://doi.org/10.1109/TSMCB.2009.2031199`.

[74] 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.

[75] 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.

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

[77] B. J. Oommen and E. Croix. On using learning automata for fast graph partitioning. In *Latin American Symposium on Theoretical Informatics*, pages 449–460. Springer, 1995.

[78] B. J. Oommen and E. Croix. Graph partitioning using learning automata. *IEEE Transactions on Computers*, 45(2):195–208, 1996.

[79] B. J. Oommen and C. Fothergill. The image examination and retrieval problem : A learning automaton-based solution. In *International Conference on Automation, Robotics, and Computer Vision (ICARCV)*, pages 21.5.1–21.5.5. IEEE, 1992.

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

[81] B. J. Oommen and J. K. Lanctot. Discretized pursuit learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, 20(4):931–938, 1990.

[82] B. J. Oommen and D. Ma. Fast object partitioning using stochastic learning automata. In *The 10th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–122. ACM, 1987.

[83] 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.

[84] B. J. Oommen and D. C. Y. Ma. Stochastic automata solutions to the object partitioning problem. *The computer journal*, 35:A105–A120, 1992.

[85] B. J. Oommen and T. D. Roberts. Continuous learning automata solutions to the capacity assignment problem. *IEEE Transactions on Computers*, 49(6):608–620, 2000.

[86] B. J. Oommen, R. S. Valiveti, and J. Zgierski. A fast learning automaton solution to the keyboard optimization problem. In *The 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 2, pages 981–990. ACM, 1990.

[87] B. J. Oommen, R. S. Valiveti, and J. R. Zgierski. An adaptive learning solution to the keyboard optimization problem. *IEEE transactions on systems, man, and cybernetics*, 21(6):1608–1618, 1991.

[88] B. J. Oommen and J. Zgierski. On breaking substitution cyphers using learning automata. In *The International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 284–293. IEA/AIE, 1991.

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

[90] B. J. Oommen and J. R. Zgierski. Keyboard optimization using genetic techniques. In *The 10th Annual International Conference on Computers and Communications*, pages 726–732. IEEE, 1991.

[91] E. J. Ormrod and M. K. Davis. *Human learning.* Merrill, 2004.

[92] G. I. Papadimitriou, A. S. Pomportsis, S. Kiritsi, and E. Talahoupi. Absorbing stochastic estimator learning algorithms with high accuracy and rapid convergence. In *The ACS/IEEE International Conference on Computer Systems and Applications*, pages 45–51. IEEE, 2001.

[93] 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.

[94] A. S. Poznyak. Learning automata in stochastic programming problems. *Automat. Telemeh*, 10:84–96, 1973.

[95] A. S. Poznyak. Investigation of the convergence of algorithms for the functioning of learning stochastic automata. *Avtomatika i Telemekhanika*, 36(1):88–103, 1975.

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

[97] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin. Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of supercomputing*, 68(1):1–48, 2014. doi: `https://doi.org/10.1007/s11227-013-1021-9`.

[98] J. J. Rodrigues and P. A. Neves. A survey on ip-based wireless sensor network solutions. *International Journal of Communication Systems*, 23(8):963–981, 2010. doi: `https://doi.org/10.1002/dac.1099`.

[99] G. Salton. *Dynamic Information and Library Processing.* ERIC, 1975.

[100] Y. Sawaragi and N. Baba. Two ϵ-optimal nonlinear reinforcement schemes for stochastic automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-4(1):126–131, 1974.

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

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

[103] A. Shirvani and B. J. Oommen. On enhancing the object migration automaton using the pursuit paradigm. *Journal of Computational Science*, 2017. doi: `https://doi.org/10.1016/j.jocs.2017.08.008`.

[104] A. Shirvani and B. J. Oommen. On utilizing the pursuit paradigm to enhance the deadlock-preventing object migration automaton, Amman, Jordan. In *The International Conference on New Trends in Computing Science*. IEEE, 9 2017. doi: `https://doi.org/10.1109/notyet`.

[105] A. Shirvani and B. J. Oommen. On utilizing the pursuit paradigm to enhance the deadlock-preventing object migration automaton, Amman, Jordan. In *The International Conference on New Trends in Computing Sciences*, pages 295–302. IEEE, 10 2017. doi: `https://doi.org/10.1109/ICTCS.2017.40`.

[106] A. Shirvani and B. J. Oommen. Partitioning in signal processing using the object migration automaton and the pursuit paradigm, Tokyo, Japan. In *International Workshop on Machine Learning for Signal Processing*. IEEE, 9 2017. doi: `https://doi.org/10.1109/MLSP.2017.8168149`.

[107] A. Shirvani and B. J. Oommen. On invoking transitivity to enhance the pursuit-oriented object migration automata. *IEEE Access*, (to appear). Accepted April 10, 2018.

[108] R. Simha and J. F. Kurose. Relative reward strength algorithms for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(2):388–398, 1989.

[109] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM (JACM)*, 32(3):652–686, 1985.

[110] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *The International Conference on Very Large Data Bases*, pages 187–198. ACM, 9 2006.

[111] M. A. L. Thathachar and K. Ramakrishnan. A hierarchical system of learning automata. *IEEE Transactions On Systems Man And Cybernetics*, 11(3):236–241, 1981.

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

[113] M. A. L. Thathachar and P. S. Sastry. A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SCM-15(1):168–175, 1985.

[114] M. A. L. Thathachar and P. S. Sastry. Estimator algorithms for learning automata. In *The Platinum Jubilee Conference on Systems and Signal Processing*, 1986.

[115] M. A. L. Thathachar and PS Sastry. A class of rapidly converging algorithms for learning automata. In *The International Conference on Systems, Man and Cybernatics*. IEEE, 1984.

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

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

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

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

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

[121] 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.

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

[123] 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.

[124] A. V. Vasilakos and G. I. Papadimitriou. Ergodic discretized estimator learning automata with high accuracy and high adaptation rate for nonstationary environments. In *The 2nd International IEEE Conference on Tools for Artificial Intelligence*, pages 245–253. IEEE, 1990.

[125] A. V. Vasilakos and G. I. Papadimitriou. A new approach to the design of reinforcement schemes for learning automata: Stochastic estimator learning algorithm. *Neurocomputing*, 7(3):275–297, 1995.

[126] R. Viswanathan and K. S. Narendra. A note on the linear reinforcement scheme for variable-structure stochastic automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 1972.

[127] R. Viswanathan and K. S. Narendra. Stochastic automata models with applications to learning systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(1):107–111, 1973.

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

[129] S. Yakowitz. Multiple hypothesis testing by finite memory algorithms. *The Annals of Statistics*, pages 323–336, 1974.

[130] A. Yazidi, O. C. Granmo, and B. J. Oommen. Service selection in stochastic environments: a learning-automaton based solution. *Applied Intelligence*, 36(3):617–637, 2012. doi: `https://doi.org/10.1007/s10489-011-0280-5`.

[131] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008. doi: `https://doi.org/10.1016/j.comnet.2008.04.002`.

[132] E. Yoneki and J. Bacon. A survey of wireless sensor network technologies: Research trends and middleware's role. *University of Cambridge*, page 45, 2005.

[133] 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.

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

[135] X. Zhang, O.-C. Granmo, and B. J. Oommen. On incorporating the paradigms of discretization and bayesian estimation to create a new family of pursuit learning automata. *Applied Intelligence*, 39(4):782–792, 2013.

[136] X. Zhang, B. J. Oommen, and O.-C. Granmo. The formal analysis for the ϵ-optimality of the family of bayesian pursuit algorithms. *Submitted to IEEE Transactions on Cybernetics*, 2014.

[137] Xuan Zhang, Ole-Christoffer Granmo, B. John Oommen, and Lei Jiao. A formal proof of the ϵ-optimality of absorbing continuous pursuit algorithms using the theory of regular functions. *Applied Intelligence*, 41:974–985, 2014.

[138] Xuan Zhang, B. John Oommen, Ole-Christoffer Granmo, and Lei Jiao. A formal proof of the ϵ-optimality of *discretized* pursuit algorithms. *submitted to Applied Intelligence*, 2014.

[139] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials*, 12(2):159–170, 2010.