On Invoking Transitivity to Enhance the *Pursuit*-oriented Object Migration Automata

Abdolreza Shirvani and B. John Oommen *Fellow, IEEE*.

***Abstract*—From the earliest studies in graph theory [2], [5], the phenomenon of transitivity has been used to design and analyze problems that can be mapped onto graphs. Some of the practical examples of this phenomenon are the "Transitive Closure" algorithm, the multiplication of Boolean matrices, the determination of Communicating States in Markov Chains etc. The use of transitivity, however, to catalyze the partitioning problems is, to our knowledge, unreported, and it is by no means trivial considering the *pairwise* occurrences of the queries in the query stream. This paper pioneers such a mechanism. In particular, we consider the Object Migrating Automaton (OMA) that has been used for decades to solve the Equi-Partitioning Problem (EPP) where W objects are placed in R partitions of equal sizes so that objects accessed together fall in to the same partition. The OMA, which encountered certain deadlock configurations, was enhanced by Gale *et al.* to yield the Enhanced OMA (EOMA). Both the OMA and the EOMA were significantly improved by incorporating into them, the recently-introduced "*Pursuit*" phenomenon from the field of Learning Automata (LA). In this paper[1] we shall show that the *Pursuit* matrix that consists of the estimates of the probabilities of the pairs presented to the LA, possesses the property of transitivity akin to the property found in graph-related problems. By making use of this observation, transitive-closure-like arguments can be made to invoke reward and penalty operations on the POMA and the PEOMA. This implies that objects can be moved towards their correct partitions *even when the system is dormant*, i.e., when the Environment *does not present* any queries or partitioning information to the learning algorithm. The results that we present demonstrate that the newly-designed transitive-based algorithms are about 20% faster than their non-transitive versions. As far as we know, these are the fastest partitioning algorithms to-date.**

Keywords*—Object Partitioning, Learning Automata, Object Migration Automaton, Partitioning-based Learning, Transitivity-enhanced Partitioning

A. Shirvani can be contacted at: School of Computer Science, Ottawa Carleton University, Ontario, Canada E-mail: ashirvan@scs.carleton.ca.

B. J. Oommen: *Chancellor's Professor*, *Fellow: IEEE* and *Fellow: IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. The author is also an Adjunct Professor with University of Agder, Grimstad, Norway. E-mail: oommen@scs.carleton.ca. His work was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

[1]We are extremely grateful to the anonymous Referees of the earlier version of this paper for their feedback. Their input improved the quality of this current version.

I. Introduction

To achieve partitioning in unknown domains, researchers have incorporated models that use hypothetical (or "abstract") 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 well-established, and has been utilized in the literature to solve 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.

Due to the poor convergence of prior OPP/EPP solutions, they were never utilized in real-life application domains, until when the first paper on the OMA was published by Oommen *et al.* [17], which 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. Since then the OMA has been incorporated to solve many real-life problems such as Cryptanalysis in [10] and [18], stochastic mapping and partitioning problems [11] and [15], Parallel and Distributed Process Mapping [12], Multi-Constraint Static Mapping in [6] and [16], Adaptive Data Structures, Quality aware applications [14], Secure Statistical Databases [3], Distributed Databases [13], Reputation Systems [9] and [24], and Cloud Computing [8] and [23] etc. We refer the reader to [20] for a detailed survey of the field and its applications.

A near-optimal solution to the OPP when the groups are of equal sizes was presented *via* the so-called Object Migrating Automaton (OMA). This problem and the OMA-based solution involves the core of this research. The OMA has been successfully applied to many real-world scenarios briefly mentioned later, and indeed, its computational overhead is insignificant. It is thus within the boundary of present-day limitations.

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

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 all the various instantiations of the OMA, the LA essentially utilized *only* the states (or the action probability vectors) associated with the various abstract objects. This assertion was true till the last few months when, the authors of this present paper observed that the information resident within the access probability matrix had been completely ignored. To place this assertion in the appropriate context, this is analogous to the family of LA which either utilized estimates of the reward probabilities or which did not. Prior instantiations of the OMA did not use these estimates.

The *Pursuit* strategy of designing LA is a special derivative of the family of estimator algorithms, which was first incorporated in the OMA in [20]. Pursuit algorithms "pursue" the currently-known best action, and increases the action probability associated with the action that possesses the largest reward estimate. The pursuit concept was first introduced by Thathachar and Sastry in [22] and its discretized version was proposed by Oommen and Lanctot [19]. Numerous Pursuit-based LA have been proposed since then (as can be seen from the survey in [19]).

Our recent results have demonstrated that invoking and incorporating the information in the so-called pursuit matrix is definitely advantageous. The first result, which utilized this was reported in [20], where we proposed the use of 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 have also incorporated this paradigm to the Enhanced OMA [21] (which resolved the deadlock scenario described in [4]). The results obtained and reported in [20] and [21] were orders of magnitude faster than their non-pursuit versions. As far as we know, this represents the state-of-the-art and we will survey these results briefly in a subsequent section.

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 wait 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 present research endeavor is to address the issues mentioned in the above observation. In the body of this work, 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.

In the partitioning problem, the physical objects are linked to one another through a physical or a metaphysical relational tie. Although the pursuit paradigm proposed in [20], [21] 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 [20].

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 enhances 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 presently.

The results obtained on benchmark Environments demonstrate that our current transitive versions is up to 100% faster than the non-transitive versions, and the current version can be nearly 90 *times* faster than the original OMA [17] in certain studied cases. By all metrics, these results are incredible.

A. Paper Organization

Section II describes the Partitioning Problem in general and provides a brief review of the field. We also discuss the Pursuit principle and how it has been utilized earlier in the Enhanced OMA (PEOMA). Section III provides the theory and the necessary background which are used later in Section V. In Section IV, we discuss how the Environment can be modeled and we provide a probabilistic representation for it. Section V incorporates the idea of transitivity and introduces the concept of *Inferred* queries as opposed to the *Real* queries presented by the Environment. In Section V-A, the convergence conditions of the Transitive PEOMA (TPEOMA) are discussed in detail. Section VI presents the simulation results for the TPEOMA, after which we study its performance. Section VIII concludes the paper and contains a brief discussion about further research directions.

II. The Generic Partitioning Problem

The ultimate goal of every partitioning algorithm is to gather the elements that "should belong together" in an unsupervised manner. First of all, we assume that there is the true unknown state of nature, i.e., Ω^*. Our aim is to try to learn a partitioning Ω^+, that is, hopefully, identical to Ω^*. Finding the proper partitioning of objects, and dividing them into relevant sub-groups, is an NP-hard problem, which has been the subject of research for more than five decades. Due to its NP-hard nature, there exist no general polynomial-time solution, and all of the reported algorithms to date, take advantage of various heuristics, which are mainly AI-based.

To assist in the task, we assume that any algorithm attempting to do the partitioning interacts with an Environment which probabilistically provides it with information about the objects that should belong together. The Environment, referred to as $\mathbb{E}$, randomly selects the initial class with probability $\frac{1}{R}$. It then picks the first object in the query from *this* class, say, q_1. The second element of the pair, q_2, is then chosen with the probability p from the same class, and with probability $1-p$ from one of the other classes uniformly, i.e., with the probability $\frac{1}{R-1}$ uniformly. We assume that $\mathbb{E}$ generates an "unending" stream of query pairs.

The Equi-Partitioning Problem (EPP) is a special case of the Object Partitioning Problem (OPP), in which there are equal number of objects in each group. A review of the previously-proposed methods for the OPP and the EPP can be found in [19] and [20].

To be specific, consider Figure 1, which represents a case in which we have 3 classes each with 3 objects.

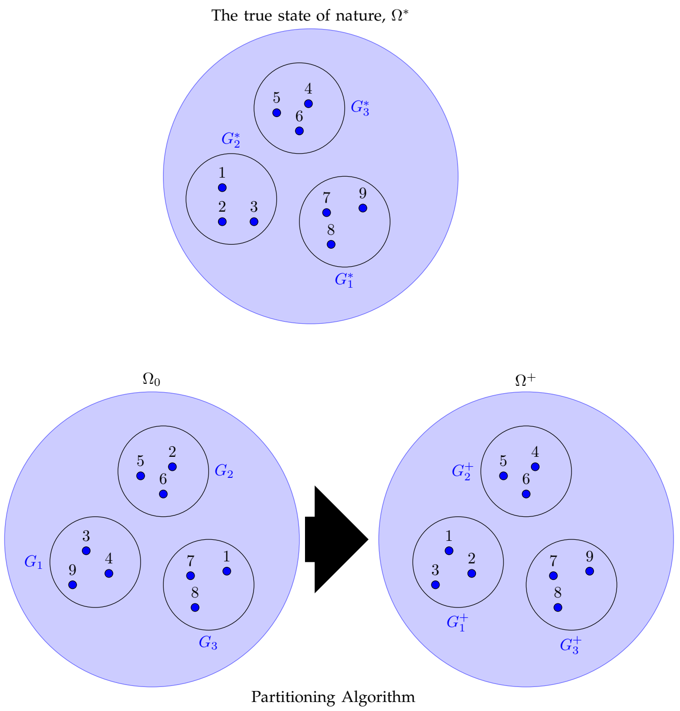


Fig. 1: The general partitioning problem.

The three classes in this example are named G_1, G_2 and G_3, and the objects inside them are represented by integers in $\{1, \cdots, 9\}$. The original distribution of the objects between the classes is shown in Figure 1, at the top. This is the true unknown state of nature, i.e., Ω^*. The AI algorithm that attempts to learn Ω^* is initialized in a purely random manner by the numbers within the range. This step is depicted at the bottom of Figure 1, and Ω_0 indicates the initial state of the algorithm. At every iteration, a pair given by $\mathbb{E}$, as discussed above, is processed by the AI algorithm, and it performs a learning step towards its convergence. The goal of the algorithm is for it to converge to a state, say Ω^+. In an optimal setting, we would hope that Ω^+ is identical to Ω^*.

A classic LA has been used as the benchmark solution for the EPP, and the corresponding methodology that uses this paradigm, is the Object Migration Automaton (OMA). It has been used in numerous application domains [19].

The performance of the OMA is highly susceptible to certain conditions, and it can even get "trapped" in a *deadlock* situation. This prevents the OMA from converging to the correct partitioning or slows down its convergence. This scenario is explained in detail in

[21]. Gale *et al.* proposed an enhanced method in [4] which we have referred to as the EOMA, in which they discussed the deadlock phenomenon and addressed its resolution, as has been done in [21] too.

The OMA and EOMA have fixed structure underlyings. They both have poor or mediocre performance when the number of partitions are large (a hard-to-solve problem) or when the Environment is difficult to learn from, i.e., when there is the presence of a high level of noise in the Environment.

A. Introducing the Pursuit Paradigm in LA

A traditional LA assumes that the actions are chosen purely based on the "state" in which the machine is. When the LA receives the feedback from the Environment, it invokes the update policy function and advances the machine to a new state. The output function takes the new state and determines the action to be taken. This strategy ignores any estimations of the Environment's reward/penalty probabilities. The families of Estimator/Pursuit LA utilize "cheap" estimates of the Environment's reward probabilities, and makes the next decision based on the action chosen, the action probability and the estimates of the reward probabilities to *prioritize* the actions. This makes the LA to converge by an order of magnitude faster. Informally, we can reinterpret it as follows: By utilizing inexpensive[2] estimates of the reward probabilities, the action-probability vector is updated not only on the basis of the Environments's response, but also based on the *ranking* of these estimates. Consequently, 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. This is referred to as the *pursuit* phenomenon.

The original OMA has had many significant applications which include (but are not limited to) Adaptive Data Structures, Secure Statistical Databases, and Distributed Database Systems (DDSs) Other applications in Reputation System [24] and Cloud Computing [8] are other emerging applications of the OMA[3].

Although the original OMA was a pioneering algorithm that had significant applications, like any other algorithm, it had 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 classes is not always necessarily achieved in the best possible manner. This is because, when the number of actions is large, the probability of receiving multiple subsequent rewards for the same action becomes very small. In the following sections we discuss and review these drawbacks.

In [20], the present authors recently incorporated the *Pursuit* phenomenon which, as in the field of LA, enhances the OMA's performance significantly. Their proposed method operates by discarding the divergent object pairs from the query stream. This can be perceived as a filtering mechanism which triggers the pursuit policy to eliminate the counter-productive queries, thus leading to a faster convergence. The resulting machine, referred to as the Pursuit OMA (POMA), has been thoroughly tested in the standard benchmark environments. Viewed from another perspective, by incorporating the pursuit paradigm, they proposed that the Pursuit concept can be used as an effective accept/reject policy, leading to a significant increase in the performance of the OMA – sometimes by a factor of 20 when compared to the original OMA [20]. This is achieved without any significant computational overhead, whenever the Environment is difficult to learn from and/or when the partitioning problem is inherently formidable.

In the *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 briefly alluded to earlier and will be explained in greater detail later. It suffices to mention that this deadlock was resolved by Gale *et al.* in [4], to create the so-called EOMA.

The present authors also integrated the pursuit concept into this enhanced version to design the PEOMA. The PEOMA utilizes the same principles used in the POMA. This again, is achieved without a significant computational overhead. The effectiveness of the pursuit paradigm in the POMA and PEOMA are discussed in details in [20] and [21] respectively.

III. 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 1 and $\mathbb{E}$'s *modus operandus*, we see that there are four common phenomena that each subgroup possesses:

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

[3]A more complete list of these applications was presented in Section I.

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 [20] and [21], 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 1. *A binary relation $\mathcal{R}$ over a set of objects W is symmetric if*

$$\forall O_i, O_j \in W : O_i \mathcal{R} O_j \iff O_j \mathcal{R} O_i. \tag{1}$$

Theorem 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 [20], [21], 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 [4], [17], it was tacit, and implemented in [20] and [21]. 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 2. *A binary relation $\mathcal{R}$ over a set of objects W is transitive if:*

$$\forall O_i, O_j, O_k \in W : (O_i \mathcal{R} O_j \wedge O_j \mathcal{R} O_k) \implies O_i \mathcal{R} O_k. \tag{2}$$

Theorem 2. *The model of $\mathbb{E}$ proposed in Section II 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 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.

A. 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[4] as:

$$\begin{aligned} \mu^*_{i,j} &= P(R_k)\cdot P(A_j|A_i)\cdot P(A_i), \forall i,j \text{ if } \langle A_i, A_j\rangle \in R_K, \\ &\qquad\qquad\qquad\qquad \text{with } k\in\{1,\cdots,R\}, \\ &= 0 \text{ otherwise,} \end{aligned} \tag{3}$$

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. The set of values of $\mu^*_{i,j}$ in Equation (3) 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 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 [20]. 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*[5] of the form:

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}^*_1 & \underline{0} & \cdots & \underline{0} \\ \underline{0} & \mathcal{M}^*_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{0} & \cdots & \cdots & \mathcal{M}^*_R \end{bmatrix} \tag{4}$$

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

[5]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.

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

$$\mathcal{M}^*_r = \begin{bmatrix} 0 & \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} \\ \frac{R}{W(W-R)} & 0 & \cdots & \frac{R}{W(W-R)} \\ \vdots & & \ddots & \vdots \\ \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} & 0 \end{bmatrix}, \tag{5}$$

and 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 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, $2\le j\le\frac{W}{R}$, and hence j can take $\frac{W}{R}-1$ possible values. Since all of these elements are equally likely, the conditional probability of j given that $i=1$ equals $\frac{1}{\frac{W}{R}-1}=\frac{R}{W-R}$. Since the total probability $P(u,v)=P(u|v).P(v)$, and since $P(O_1)=\frac{1}{W}$, 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 analogously.

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 (5), and that the sum of all the probabilities in $\mathcal{M}^*$, given in Equation (4), is unity. This concludes the first part of the proof.

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) must be 0. Otherwise, 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, 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.

IV. 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 [20], 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 2 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 2. The transitivity of $\mathbb{E}$ can thus be easily inferred, as we shall do in Theorem 2.

The next subsection formalizes the phenomenon.

A. Modeling the Noisy Environment and its Transitivity

Theorem 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 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 [20]. 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*[6]. 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 (6):

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}^*_1 & \underline{\theta} & \cdots & \underline{\theta} \\ \underline{\theta} & \mathcal{M}^*_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{\theta} & \cdots & \cdots & \mathcal{M}^*_R \end{bmatrix}, \tag{6}$$

where $\underline{\theta}$ and $\mathcal{M}^*_r$s are specified as per Equation (7) and Equation (8) respectively. In the above,

$$\underline{\theta} = \theta_o \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & 1 \end{bmatrix}, \tag{7}$$

and

$$\mathcal{M}^*_r = \theta_d \cdot \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix}, \tag{8}$$

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 (8) and Equation (7) relations.

To prove the above, as in the case of Theorem 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[7] with probability $1 - \theta$. Since $\mathcal{M}^*_1$ is symmetric, all the entries along the diagonal are zero and the off-diagonal entries represent the within-class probabilities of $\langle O_i, O_j \rangle$ where $2 \le j \le \frac{W}{R}$. Since all the $[1, j]$ entries of $\mathcal{M}^*_1$ are equally likely[8], the form of Equation (8) 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 (8).

If A_2 belongs to a class distinct from A_1, as opposed to the case of Theorem 3, 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

[6]As in the case of [20] and [21], 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.

[7]This must be contrasted with Theorem 3 where $\mathbb{E}$ cannot choose the second element from any other class.

[8]The equally-likely condition is a necessity to demonstrate transitivity. It was not required in the case of the [20] and [21].

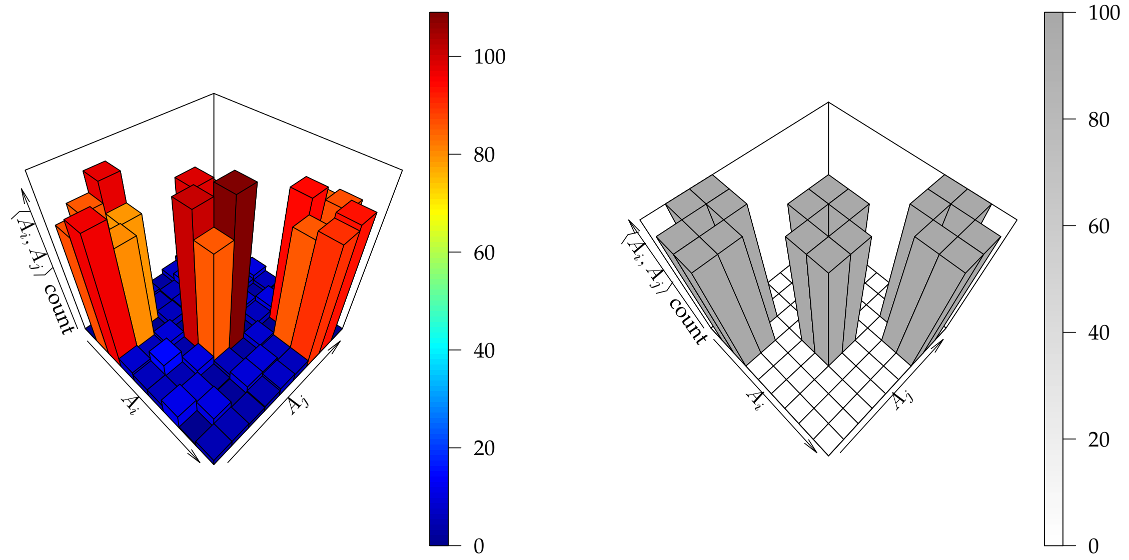


Fig. 2: The estimation of $\mathcal{M}^*$ which allows us to invoke the pursuit concept. The number of partitions, R, in this case, is 3 and $p = 0.8$. The figure to the left represents the joint probabilities of the objects. The figure to the right is the binary valued representation as in the text.

$\mathcal{M}_1^*$ 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 (7) 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:

$$\theta_d(\frac{W}{R} - 1) + \theta_o(W - \frac{W}{R}) = 1, \tag{9}$$

whence, we can see that θ_d has the form:

$$\theta_d = \frac{1 - \theta_o(W - \frac{W}{R})}{\frac{W}{R} - 1}. \tag{10}$$

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 (10). The reader will observe that θ_d in Equation (10), 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 2, without explicitly maintaining the matrix $\mathcal{Q}$. This is explained in the next section.

V. 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 [20] and [21]. The first phase is identical to the PEOMA which is given in Algorithm 2. 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 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[9].

The next phase will consider how the property of transitivity can be taken advantage of. The reader will observe that Theorem 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 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[10]. 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 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

[9]The reader can refer to [20] for the details on how we can specify the values of τ and κ as far as he PEOMA is concerned.

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

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 below.

A. Remarks Regarding the TPEOMA

The pursuit matrix collects the likelihood of objects occurring together in the matrix $\mathcal{M}^*$, defined in Theorem 3 and Theorem 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 2, $\mathcal{M}^*$ 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 1) below.

Algorithm 1 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 $\dfrac{1}{W^2 - W}$.
- 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 $(k-1)N + 1 \le \xi_i \le kN$ then object O_i is assigned to α_k.

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

The reader should specifically note the following:

1) First of all, the TPEOMA is design 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 [21].
5) Finally, with regard to the *Real* and *Inferred* queries, we invoke the reward/penalty function (line 9 and 12 in Algorithm 1, and lines 6 and 8 in Algorithm 2) 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 1 and lines 6 and 8 of Algorithm 2, avoiding code duplication.

Algorithm 2 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 $\xi_i$ div $N = \xi_j$ div $N$ then    ▷ The partitioning is rewarded
3:         Call ProcessRewardEOMA($\{\xi_p\}, A_i, A_j$)
4:     else    ▷ The partitioning is penalized
5:         Call ProcessPenaltyEOMA($\{\xi_p\}, A_i, A_j$)
6:     end if
7:     return $\{\xi_p\}$
8: end
```

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

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

VI. 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 [21] 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 [21]. The convergence conditions were also identical to the ones specified in [21], 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 [21]. 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 $\kappa > \left[\left(\frac{W}{R}\right)^2 - \frac{W}{R} \right] \times R$.

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

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 $\xi_i$ mod $N \neq 0 \wedge \xi_j$ mod $N \neq 0$ then    ▷ Both are in internal states
3:         $\xi_i = \xi_i + 1$
4:         $\xi_j = \xi_j + 1$
5:     else if $\xi_i$ mod $N \neq 0$ 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 $\xi_j$ mod $N \neq 0$ 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
```

The simulation results are given in Table II for the PEOMA and in Table II for the TPEOMA, both of 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 level 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* uses the *same* pursuit matrix, and is unrivaled by any other reported variations of the OMA.

By utilizing both the internal improvements as well as the pursuit concept in extracting transitivity rela-

TABLE I: 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 *and accurately* converged.
In all experiments, the number of states of the POMA is set to 10.
$POMApX$: $\mathcal{X}$ 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 $\kappa^* = R\left[(\frac{W}{R})^2 - \frac{W}{R}\right]$.

TABLE II: 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 *and accurately* converged.
In all experiments, the number of states of the TPEOMA is set to 10.
$TPEOMApX$: $\mathcal{X}$ 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 $\kappa^* = R\left[(\frac{W}{R})^2 - \frac{W}{R}\right]$.

tions from $\mathbb{E}$, the TPEOMA is capable of operating in extremely complex (highly noisy) environments, and solving difficult problems even faster than PEOMA.

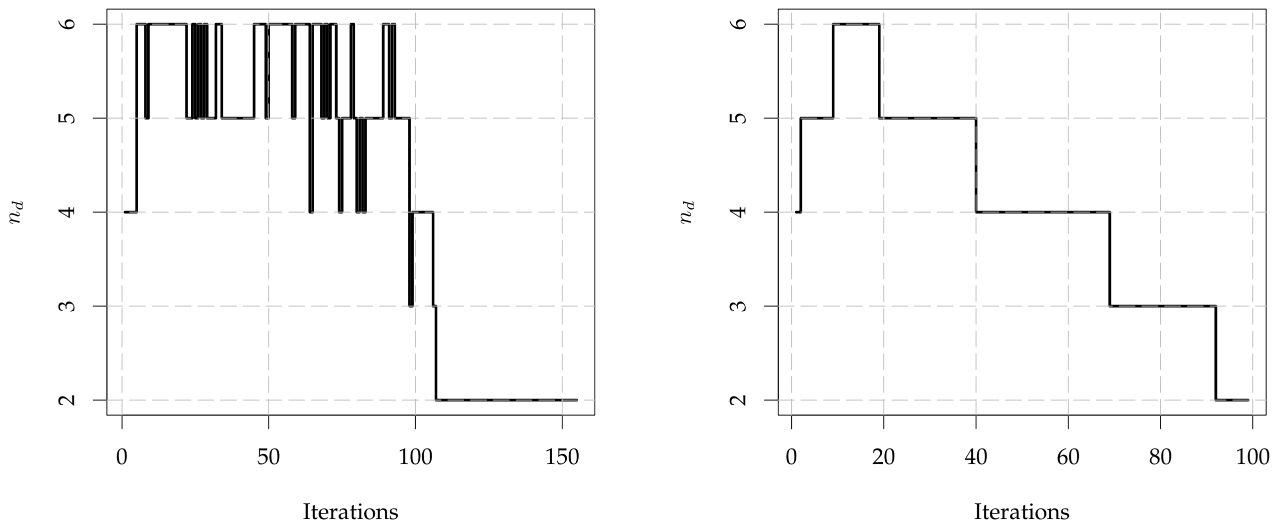
In comparison with the original PEOMA in [21], the Algorithm 1 also updates the states of the *inferred* objects which are in the transitivity relation with the received object pair. In summary, the introduced modification enhances the PEOMA's performance by introducing a linear overhead $\mathcal{O}(W)$ by looking in to a row of W objects and choosing the objects which are above the 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.

VII. Discussion

By monitoring the execution of the algorithm, it is obvious that the argument presented in Theorem 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 [20] and [21], this paper 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.



(a) Single run.

(b) Ensemble of 100 runs.

Fig. 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. Here, $W = 18$, $R = 2$ and $p = 0.9$.

Thus, the reader will observe that 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 [21]), 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.

It is pertinent to query the robustness of the proposed schemes[11]. To consider this, rather that merely refer to the tables, we consider the corresponding convergence graphs. The convergence graph for a single experiment of the TPEOMA is depicted in Figure 3a. 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 3a. However, if one takes the ensemble average of this metric over 100 experiments the performance is much more monotonic in behavior (with almost no random fluctuations), demonstrating the robusness over a large set of experiments. This is clear from Figure 3b. The reader should observe the considerable performance that is gained by a very little additional computational cost. Again, by comparing Tables I and II, although the gain is not significant for simple problems and easy Environments, it becomes remarkably high for complex partitioning experiments.

[11]We are grateful to the anonymous Referees of the earlier version of this paper who raised this query.

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 *nine* times faster than the original OMA.

VIII. 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 $\mathbb{E}$ 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 *nine* 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 artificially generated by $\mathbb{E}$ after enough number of iterations. By taking into account such an intuition, one can incorporate the hidden ties among the objects, and produce virtual reward and penalty responses based on these *Inferred* queries. This is able to significantly increase the convergence ratio in complex partitioning problems. We thus see that a quantitative analysis of the pursuit matrix in the EPP, in conjunction with interpreting them qualitatively, opens up new avenues in the research of the EPP.

With regard to future work, we foresee that it would be profitable to address broader and more complicated sets of problems, such as the Unbalanced Partitioning Problem (UPP), and to consider application domains that map onto the EPP and the UPP for which the TPEOMA provides an efficient solution.

References

[1] 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.

[2] N. Biggs. *Algebraic Graph Theory*. Cambridge university press, 1993.

[3] 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.

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

[5] C. Godsil and G. F. Royle. *Algebraic Graph Theory*, volume 207. Springer Science & Business Media, 2013.

[6] B. Hendrickson and T. G Kolda. Graph partitioning models for parallel computing. *Parallel computing*, 26(12):1519–1534, 2000.

[7] 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.

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

[9] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[10] A. Kendhe and H. Agrawal. A survey report on various cryptanalysis techniques. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(2), 2013.

[11] J. J. Leonard and H. J. S Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Robotics Research*, pages 169–176. Springer, 2000.

[12] J. Li and J. Zhangt. Using estimation of distribution algorithm to coordinate decentralized learning automata for meta-task scheduling. In *The IEEE Congress on Evolutionary Computation*, pages 2077–2084. IEEE, 2014. doi: https://doi.org/10.1109/CEC.2014.6900426.

[13] 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.

[14] S. B. Musunoori and G. Horn. A fixed-structure learning automation solution to the quality aware application service configuration in a grid environment. In *The International Conference on Parallel and Distributed Computing Systems*, pages 295–302. IEEE, Nov. 2005. doi: https://doi.org/10.1109/ICTCS.2017.40.

[15] S. B. Musunoori and G. Horn. Intelligent ant-based solution to the application service partitioning problem in a grid environment. In *The 6th International Conference on Intelligent Systems Design and Applications*, volume 1, pages 416–424. IEEE, 2006.

[16] Temel Öncan. A survey of the generalized assignment problem and its applications. *Information Systems and Operational Research (INFOR)*, 45(3):123–141, 2007.

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

[18] 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.

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

[20] 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.

[21] 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, Oct. 2017. doi: https://doi.org/10.1109/ICTCS.2017.40.

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

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

[24] 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.