The Power of the "Pursuit" Learning Paradigm in the Partitioning of Data

Abdolreza Shirvani, B. John Oommen

The Power of the "Pursuit" Learning Paradigm in the Partitioning of Data

Abdolreza Shirvani[1] and B. John Oommen[1,2]

[1] School of Computer Science, Carleton University, Ottawa, Canada
[2] Centre for Artificial Intelligence Research, University of Agder, Grimstad, Norway

Abstract. Traditional Learning Automata (LA) work with the understanding that the actions are chosen purely based on the "state" in which the machine is. This *modus operandus* completely ignores any estimation of the Random Environment's (RE's) (specified as $\mathbb{E}$) reward/penalty probabilities. To take these into consideration, Estimator/Pursuit LA utilize "cheap" estimates of the Environment's reward probabilities to make them converge by an order of magnitude faster. This concept is quite simply the following: Inexpensive estimates 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 Environment's response alone, but also based on the ranking of these estimates. While this phenomenon has been utilized in the field of LA, until recently, it has not been incorporated into solutions that solve partitioning problems. In this paper[3], we will submit a complete survey of how the "Pursuit" learning paradigm can be and has been used in Object Partitioning. The results demonstrate that incorporating this paradigm can hasten the partitioning by a order of magnitude.

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

1 Introduction

The Pursuit Concept in LA: Absolutely Expedient LA are absorbing and there is always a small probability of them not converging to the best action. Thathachar and Sastry realized this phenomenon and proposed to use Maximum Likelihood Estimators (MLEs) to hasten the LA's convergence. Such an MLE-based update method would utilize estimates of the reward probabilities in the update equations. At every iteration, the estimated reward vector was also used to update the action probabilities, instead of updating it based only on the RE's feedback. In this way, the probabilities of choosing the actions with higher reward estimates were increased, and those with lower estimates were significantly reduced, using which they proposed the family of estimator algorithms.

[3] The second author gratefully acknowledges the partial support of NSERC, the Natural Sciences and Engineering Council of Canada.

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 increase the action probability associated with *this* action. The pursuit concept was first introduced by Thathachar *et al*, and the corresponding LA was proven to be ϵ-optimal. Its discretized version was proposed by Lanctot *et al* in [7], who also discretized the original estimator algorithm. Agache *et al* [10] then analyzed all the four linear combinations, i.e., the L_{RI} and L_{RP} paradigms.

The Object Partitioning Problem (OPP): 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 $\{G_i^*\}$ is unknown, and the elements in the subsets fall together based on some criteria which may be mathematically formulated, or may even be ambiguous. These objects are now presented to a learning algorithm, 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/tuples of $\mathcal{A}$ are presented to the learning algorithm known. This problem is known to be NP-hard [9]. 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. To resolve this, it is possible to explore all partition combinations, use a ranking index, and to thereafter, report the best plausible partition. The goal of the OPP is to identify the *best* or *most likely realizable* partitioning. This requires the AI algorithm to perceive the semantic physical world aspects of the objects, and to then make local decisions based on the best partition in the *abstract* domain [4, 5].

Real *versus* Abstract Objects: If there exists a mutual relation between the real objects in the semantic domain $\mathcal{A}$, 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 $\mathcal{O}$ map onto the partitions of the real objects in $\mathcal{A}$ so as to mimic the state-of-nature. Thus, while we operate on the abstract objects in $\mathcal{O}$, the objects in $\mathcal{A}$ are not necessarily moved because they constitute real-life objects which cannot be easily moved. A special case of the OPP is the Equi-Partitioning Problem (EPP) in which all the partitions are equi-sized.

The Object Migrating Automation (OMA): Due to the poor convergence of prior OPP/EPP solutions, they were never utilized in real-life applications. The introduction of an LA-based partitioning algorithm, the OMA, (explained in Section 2) made real-life applications possible. The OMA resolved the EPP both efficiently and accurately. This solution is regarded as a benchmark for the EPP. Indeed, since 1986[4], it has been applied to variety of real-life problems and domains which include keyboard optimization, image retrieval, distributed computing, graph partitioning, the constraint satisfaction problems, cryptanalysis, reputation systems, parallel and distributed mapping etc.

[4] The bibliography in this paper is necessarily limited. The majority of the present results *very briefly* summarize the results in the Ph.D. thesis of the First Author.

The Intent of this Paper: Although the "Pursuit" learning paradigm has been utilized in the theory and applications of LA as fundamental *machines*, until recently, it has not been incorporated into solutions that solve partitioning problems. The goal of this paper is to submit a comprehensive survey of how this paradigm can be used in Object Partitioning, and to optimize various versions of the OMA. We also include simulation results on benchmark environments that demonstrate the advantages of incorporating it into the respective machines.

2 The Object Migration Automata

The OMA is a fixed structure LA designed to solve the EPP. It is defined as a quintuple with R actions[5], 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 number, and can move from one state to another, or migrate from one group to another. If the abstract object O_i is in state ξ_i belonging to a group α_k, we say that O_i is assigned to class 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 are jointly rewarded by $\mathbb{E}$, the Environment. Otherwise, they will be penalized. We formalize the movements of $\{O_i\}$ on reward/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 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) in [11][6].
- **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:
  1. If $\xi_i \neq \phi_{kN}$ and $\xi_j \neq \phi_{mN}$, we move O_i and O_j one state toward ϕ_{kN} and ϕ_{mN}, respectively, as shown in Figure 3.2 (b) in [11].
  2. 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) in [11]. 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) in [11].

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

[6] The OMA's algorithms/figures are in [11], and omitted here in the interest of space.

3. 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) in [11].

To asses the partitioning accuracy and 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 OMA's goal is to migrate the objects between its classes, using the incoming queries. $\mathbb{E}$ is characterized by three parameters: (a) W, the number of objects, (b) R, the number of partitions, and (c) a probability 'p' quantifying how $\mathbb{E}$ pairs the elements in the query.

Every query presented to the OMA by $\mathbb{E}$ consists of two objects. $\mathbb{E}$ 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.

The results of the simulations are given in Table 1, where in $OMAp\mathcal{X}$, $\mathcal{X}$ refers to the probability specified above, W, is the number of objects, W/R is the number of objects per class, and R is the number of classes. 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. Also, the OMA's convergence for a single run and for an ensemble of runs display a monotonically decreasing pattern (with time) for the latter.

Table 1. Experimental results for the OMA done for an ensemble of 100 experiments in which we have only included the results from experiments where convergence has occurred.

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

3 Developing the Pursuit Concept: The Environment

In an "un-noisy" Environment, we can denote the actual value of the relation between A_i and A_j (for $k \in \{1, \cdots, R\}$) by the quantity $\mu^*(i,j)$, expressed as:

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

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. Since $\mathbb{E}$ chooses the elements of the pairs from the other groups uniformly, with a possible re-numbering operation, the matrix $\mathcal{M}^* = [\mu^*(i,j)]$ is a *block-diagonal* matrix given by Eq. (1).

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}_1^* & \underline{0} & \cdots & \underline{0} \\ \underline{0} & \mathcal{M}_2^* & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{0} & \cdots & \ldots & \mathcal{M}_R^* \end{bmatrix} \tag{1}$$

where $\underline{0}$ represents a square matrix containing only 0's.

Theorem 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:*

$$\mathcal{M}_r^* = \begin{bmatrix} 0 & \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} \\ \frac{R}{W(W-R)} & 0 & \cdots & \frac{R}{W(W-R)} \\ \vdots & & \ddots & \vdots \\ \frac{R}{W(W-R)} & \cdots & \frac{R}{W(W-R)} & 0 \end{bmatrix} \tag{2}$$

Proof. The proof of the theorem is omitted here. It is found in [11]. □

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

$$\mathcal{M}^* = \begin{bmatrix} \mathcal{M}_1^* & \underline{\theta} & \cdots & \underline{\theta} \\ \underline{\theta} & \mathcal{M}_2^* & & \vdots \\ \vdots & & \ddots & \vdots \\ \underline{\theta} & \cdots & \ldots & \mathcal{M}_R^* \end{bmatrix} \tag{3}$$

where $\underline{\theta}$ and $\mathcal{M}_r^*$s are specified as per Equations (4) and (5).

Theorem 2. *In the presence of noise in $\mathbb{E}$, the entries of the pair $\langle A_i, A_j \rangle$ can be selected from two different distinct classes, and hence the matrix specifying the probabilities of the accesses of the pairs obeys Equation (3), where:*

$$\underline{\theta} = \theta_o \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & 1 \end{bmatrix}, \tag{4}$$

$$\mathcal{M}_r^* = \theta_d \cdot \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix}, \tag{5}$$

where, $0 < \theta_d < 1$ *is the coefficient which specifies the accuracy of* $\mathbb{E}$*, and* θ_o *is related to* θ_d *as* $\theta_d = \frac{1-\theta_o(W-\frac{W}{R})}{\frac{W}{R}-1}$.

Proof. The proof of the theorem is omitted here and found in [11].

3.1 The Design and Results of the POMA

In a real world scenario, since $\mathbb{E}$'s true statistical model is unknown, the expressions in Equations (1) and (2) can only be estimated through observing a set of queries. In the presence of noise though, we need to devise a measurable quantity which makes the algorithm capable of recognizing divergent pairs.

Observe that whenever a real query $\langle A_i, A_j \rangle$ appears, we will be able to obtain a simple ML estimate of how frequently A_i and A_j are accessed concurrently. Clearly, by virtue of the Law of Large Numbers, these underlying estimates will converge to the corresponding probabilities of $\mathbb{E}$ actually containing the elements A_i and A_j in the same group. As the number of queries processed become larger, the quantities inside $\mathcal{M}_i^*$ will become significantly larger than the quantities in each of the $\underline{\theta}$ matrices. From the plot of these estimates [11], one will observe that the estimates corresponding to the matrix $\mathcal{M}_i^*$ have much higher values than the off-diagonal entries. This implies that these off-diagonal entries represent *divergent* queries which move the objects away from their accurate partitions.

Intuitively, the pursuit concept for the OPP can be best presented by a matrix of size $W \times W$ where every entry will capture the same statistical measure about the stream of the input pairs. For the sake of simplicity, we use a simple averaging and denote this matrix by $\mathcal{P}$. Every block represents a pair and the height of the block is set to the frequency count of the reciprocal pair. To obtain the average frequency of each pair, we let the OMA iterate for a sufficient time, say J iterations, and at every incident we update the value of the matrix $\mathcal{P}$ respectively. In this way, at the end of the J-th iteration, we have simply estimated the frequency of each pair. At this point, by observing the values of the matrix, the user can determine an appropriate threshold ($\tau > 0$) to be adopted as the accept or reject policy for any future occurrence of this particular pair of objects. If we permit the algorithm to collect a large enough number of pairs, we see that $\exists \theta^* \mid \forall \theta_o \leq \theta^*$, and that $\forall\, i, j :\ \mu_{i,j} \gg \theta^*_{i,j}$.

If we utilize a user-defined threshold, τ, (which is reasonably close to 0), we will be able to compare every estimate to τ and make a meaningful decision about

the identity of the 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 on Page 74 of [11] in which every query which is inferred to be divergent is ignored. Otherwise, one invokes the Reward and Penalty functions of the original OMA algorithm. The issue of determining the parameters of the POMA algorithm are detailed in [11], and omitted here in the interest of space.

In the initial phase of the algorithm, the estimates for the queries are unavailable. Thus, it only makes sense to consider every single query and to process them using the OMA's Reward and Penalty functions. Since the objects in each class are equally-likely to happen and the classes are equi-probable, $k \geq \left[\left(\frac{W}{R}\right)^2 - \frac{W}{R} \right] \times R$, is chosen as the lower-bound of the number of iterations for any meaningful initialization.

We have compared our results with those presented in [5] and those reported for the original OMA for various values of R and W. The number of states in every action was set to 10, and the convergence was expected to have taken place as soon as all the objects in the POMA fell within the last two internal states. The results (specified using the same notation as in Table 1) obtained are outstanding and are summarized in Table 2. The simulation results are based on an ensemble of 100 runs with different uncertainty values, (i.e., values of p).

Table 2. Experimental results for the POMA approach done for an ensemble of 100 runs.

W	W/R	R	POMAp9	POMAp8	POMAp7
4	2	2	$(2, 25)$	$(3, 30)$	$(3, 38)$
6	2	3	$(4, 44)$	$(4, 52)$	$(5, 67)$
-	3	2	$(20, 65)$	$(22, 77)$	$(24, 106)$
9	3	3	$(44, 105)$	$(70, 148)$	$(85, 169)$
12	2	6	$(10, 88)$	$(12, 103)$	$(20, 173)$
-	3	4	$(77, 166)$	$(105, 205)$	$(292, 462)$
-	4	3	$(328, 417)$	$(228, 372)$	$(202, 487)$
-	6	2	$(1563, 1836)$	$(945, 1091)$	$(1088, 1395)$
15	3	5	$(112, 213)$	$(142, 274)$	$(179, 315)$
-	5	3	$(1534, 1655)$	$(766, 998)$	$(556, 931)$
18	2	9	$(20, 151)$	$(26, 161)$	$(29, 566)$
-	3	6	$(245, 410)$	$(198, 417)$	$(226, 395)$
-	6	3	$(3146, 3270)$	$(2182, 2371)$	$(1145, 1542)$
-	9	2	$(5500, 5621)$	$(5064, 5523)$	$(4104, 4711)$

To observe the efficiency of the POMA, consider an easy-to-learn Environment of 6 groups with 2 objects in each group and where $p = 0.9$. It took the OMA 599 iterations to converge. As opposed to this, the POMA converged in only 69 iterations, which represents a *ten-fold* improvement. On the other hand, given a difficult-to-learn Environment with 12 objects in 2 groups, the OMA needs $6,506$ iterations to converge. The POMA required only $2,112$ iterations to converge, which is more than a *three-fold* improvement.

4 Enhanced OMA (EOMA)

The learning of an enhanced LA proposed by Gale *et al.* [5], is based on the same principles of the OMA and leads to the Enhanced OMA (EOMA). They introduced three enhancements to improve its efficiency and speed as below:

1. **Initial Boundary State Distribution**: All of the objects are initially distributed at the respective boundary states of their respective classes;
2. **Redefinition of Internal States**: They diminished the vulnerability of the *convergence criterion* of the OMA by redefining the internal state to include "the *two* innermost states of each class", rather than a single innermost state.
3. **Breaking the Deadlock**: The original OMA possesses a deadlock-prone infirmity (please see Section 4.3 of [11]) in which the machine can cycle between two identical configurations by virtue of a sequence of query pairs. Thus is especially evident in noise-free Environments. The EOMA remedies this as follows. 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 the OMA.

The simulation results obtained by introducing all of the three above-mentioned modifications are given in Table 3. 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 [5] with the EOMA's simulation results. These are the results displayed in Table 3. 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 increase, the problem becomes increasingly complex to solve. The reader will easily observe the advantages gleaned by the above three modifications in the EOMA, by comparing Tables 1 and 3.

The convergence of the EOMA with respect to time starts with a large number of objects which are located in random partitions. This number steadily decreases with time to a very small value. This graph is not monotonic for any given experiment. But from the perspective of an ensemble, the performance is much more monotonic in behavior.

Table 3. Experimental results for the *Enhanced* OMA (EOMA) done for an ensemble of 100 runs.

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

5 Enhancing the EOMA with a Pursuit Paradigm

The methodology by which we incorporated the pursuit concept into the OMA required us to formally model noise-free and noisy queries in Section 3. However, this is the precise dilemma that the EOMA faces. On the one hand, it would be advantageous, from a partitioning perspective, to have a noise-free Environment. However, it is precisely such noise-free Environments that lead to deadlock situations. Consequently, an attempt to elevate a noisy Environment to become noise-free would only defeat the purpose by exaggerating deadlock scenarios. 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 "Pursuit" paradigm, explained below for this specific setting.

To design the PEOMA, as before, we again 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 all brevity, the PEOMA utilizes the exact same Pursuit principles explained in Section 3.1. Essentially, the EOMA which mitigates the "deadlock" situation is now augmented with the Pursuit concept, and thus:

- 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 experimental results compared the PEOMA with the EOMA, and we were able to show how the PEOMA out-performed the EOMA and the OMA. The results are given in Table 4 which uses the same notation and for the same settings as in the Tables 1 and 2. One can also compare its performance with the results presented in [5] and those reported in Table 3 for various values of R and W and in different Environments. The number of states in every action was set to be 10 as in Table 4. The convergence condition was also identical to the one specified in Section 4, and was assumed to have taken place when 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 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 was 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 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.

Table 4. Experimental results for the PEOMA approach done for an ensemble of 100 runs.

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

6 Cohesiveness in the EPP: The Transitive PEOMA

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. They can be seen to be:

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.

We first formalize the partitioning problem's symmetry and transitivity properties proven in [11].

Theorem 3. *The model of* $\mathbb{E}$ *and the solution invoked by any pursuit-based paradigm of the EPP possess the property of symmetry.*

Theorem 4. *The model of* $\mathbb{E}$ *proposed for the EPP possesses the property of transitivity from a probabilistic perspective.*

Since $\mathbb{E}$ is transitive, our aim is now to have the LA infer this transitivity and to further enhance the PEOMA. Indeed, 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. Without going into the explicit details (omitted due to space limitations), this is essentially done by invoking the assertion: $\forall O_i, O_j, O_k \in W : (O_i \mathcal{R} O_j \wedge O_j \mathcal{R} O_k) \implies O_i \mathcal{R} O_k$. This leads us to the Transitive PEOMA (TPEOMA). The experimental results for the PEOMA are given in Table 5 for the same settings as in the previous tables.

By way of example, if the TPEOMA is compared with the previously best-reported algorithm, the PEOMA reported in Section in 5, 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 certainly the fastest partitioning algorithm reported to date, and its behavior is monotonically decreasing for an ensemble of many experiments. The reader should observe the considerable performance that is gained by a very little additional computational cost. Again, by comparing Tables 4 and 5, one observes that although the gain is not significant for simple problems and easy Environments, it becomes remarkably high for complex partitioning experiments.

Table 5. Experimental results for the TPEOMA approach done for an ensemble of 100 runs.

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

7 Conclusions

In this paper we have shown how we can utilize the "Pursuit" concept to enhance solutions to the general problem of partitioning. Unlike traditional Learning Automata (LA), which work with the understanding that the actions are chosen purely based on the "state" in which the machine is, the "Pursuit" concept has been used to estimate the Random Environment's (RE's) reward probabilities and to take these into consideration to design Estimator/Pursuit LA. They, utilize "cheap" estimates of the Environment's reward probabilities to make them converge by an order of magnitude faster. This is achieved by using inexpensive estimates of the reward probabilities to rank the actions. Thereafter, when the action probability vector has to be updated, it is done not on the basis of the Environment's response alone, but also based on the ranking of these estimates.

In this paper we have shown how the "Pursuit" learning paradigm can be and has been used in Object Partitioning. The results demonstrate that incorporating this paradigm can hasten the partitioning by a order of magnitude. This paper comprehensively describes all the Object Migration Automaton (OMA)-related machines to date, including the Enhanced OMA [5]. It then incorporates the Pursuit paradigm to yield the Pursuit OMA (POMA), the Pursuit Enhanced OMA (PEOMA) and the Pursuit Transitive Enhanced OMA (PTOMA).

Apart from the schemes themselves, the papers reports the experimental results that have been obtained by testing them on benchmark environments.

References

1. Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.
2. Norman 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. Eugene C. Freuder. The object partition problem. *Vision Flash*, -(4), 1971.
5. W. Gale, S. Das, and C. T. Yu. Improvements to an algorithm for equipartitioning. *Computers, IEEE Transactions on*, 39(5):706–710, 1990.
6. A. Jobava. Intelligent traffic-aware consolidation of virtual machines in a data center. Master's thesis, University of Oslo, 2015.
7. J. K. Lanctot and B. J. Oommen. Discretized estimator learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1473–1483, 1992.
8. A. S. Mamaghani, M. Mahi, and M. Meybodi. A learning automaton based approach for data fragments allocation in distributed database systems. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 8–12. IEEE, 2010.
9. B. J. Oommen and D. C. Y. Ma. *Stochastic automata solutions to the object partitioning problem*. Carleton University, School of Computer Science, 1986.
10. B. J. Oommen and M. Agache. Continuous and discretized pursuit learning schemes: various algorithms and their comparison. *Part B: Cybernetics, IEEE Transactions on Systems, Man, and Cybernetics*, 31(3):277–287, 2001.
11. A. Shirvani. *Novel Solutions and Applications of the Object Partitioning Problem*. PhD thesis, Carleton University, Ottawa, Canada, 2018.
12. 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.
13. 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.