Applied Intelligence (2020) 50:2916–2931
https://doi.org/10.1007/s10489-020-01657-9

A team of pursuit learning automata for solving deterministic optimization problems

Anis Yazidi[1] · Nourredine Bouhmala[1] · Morten Goodwin[1]

Published online: 14 April 2020

Abstract
Learning Automata (LA) is a popular decision-making mechanism to "determine the optimal action out of a set of allowable actions" [1]. The distinguishing characteristic of automata-based learning is that the search for an optimal parameter (or decision) is conducted in the space of probability distributions defined over the parameter space, rather than in the parameter space itself [2]. In this paper, we propose a novel LA paradigm that can solve a large class of deterministic optimization problems. Although many LA algorithms have been devised in the literature, those LA schemes are not able to solve deterministic optimization problems as they suppose that the environment is stochastic. In this paper, our proposed scheme can be seen as the counterpart of the family of pursuit LA developed for stochastic environments [3]. While classical pursuit LAs can pursue the action with the highest reward estimate, our pursuit LA rather pursues the collection of actions that yield the highest performance by invoking a team of LA. The theoretical analysis of the pursuit scheme does not follow classical LA proofs, and can pave the way towards more schemes where LA can be applied to solve deterministic optimization problems. Furthermore, we analyze the scheme under both a constant learning parameter and a time-decaying learning parameter. We provide some experimental results that show how our Pursuit-LA scheme can be used to solve the Maximum Satisfiability (Max-SAT) problem. To avoid premature convergence and better explore the search space, we enhance our scheme with the concept of artificial barriers recently introduced in [4]. Interestingly, although our scheme is simple by design, we observe that it performs well compared to sophisticated state-of-the-art approaches.

Keywords Distributed learning · Learning automata · Deterministic optimization

1 Introduction

Learning Automata (LA) have been used in systems that have incomplete knowledge about the Environment in which they operate [1, 5–11]. The learning mechanism attempts to learn from a *stochastic Teacher* which models the environment. In his pioneering work, Tsetlin [12] attempted to use LA to model biological learning. In general, a random action is selected based on a probability vector, and these action probabilities are updated based on the observation of the Environment's response, after which the procedure is repeated.

The term "Learning Automata" was first publicized and rendered famous in the survey paper by Narendra and Thathachar. The goal of LA is to "determine the optimal action out of a set of allowable actions" [1]. The distinguishing characteristic of automata-based learning is that the search for the optimizing parameter vector is conducted in the space of probability distributions defined over the parameter space, rather than in the parameter space itself [2].

Concerning applications, the entire field of LA and stochastic learning has had a myriad of applications [5–7, 9, 10], which (apart from the many applications listed in these books) include solutions for problems in network and communications [13–16], network call admission, traffic control, quality of service routing, [17–19], distributed scheduling [20], training hidden Markov models [21], neural network adaptation [22], intelligent vehicle control [23], and even fairly theoretical problems such as graph partitioning [24]. In addition to these fairly generic applications, with a little insight, LA can be used to assist in solving (by, indeed, learning the associated parameters) the stochastic resonance problem [25], the stochastic sampling problem in computer graphics [26], the problem of determining roads in aerial images by

✉ Anis Yazidi
anis.yazidi@oslomet.no

[1] Department of Computer Science, Oslo Metropolitan University, Oslo, Norway

using geometric-stochastic models [27], and various location problems [28]. Similar learning solutions can also be used to analyze the stochastic properties of the random waypoint mobility model in wireless communication networks [29], achieve spatial point pattern analysis codes for GISs [30], digitally simulate wind field velocities [31], interrogate the experimental measurements of global dynamics in magneto-mechanical oscillators [32], and to analyze spatial point patterns [33]. LA-based schemes have already been utilized to learn the best parameters for neural networks [22], optimizing QoS routing [19], and bus arbitration [14] – to mention a few other applications.

Although many LA algorithms have been devised in the literature, those LA schemes are not able to solve deterministic optimization problems as they suppose that the environment is stochastic. In other words, classical LA schemes resort to the assumption that the response of the environment to the same action or set of actions is stochastic. However, in deterministic optimization problems, this is not the case as the output, which is the response of the environment is a deterministic function of the input. There have been many studies that resort to a team of LA for solving optimization problems where the objective function is noisy. Examples of those works include noise-tolerant learning of half-spaces [34] and nonlinear fractional knapsack problem [35]. The latter stream of works show that pursuit LA is a viable solution when the objective function is noisy. However, when the objective function to optimize is deterministic, .i.e. non-noisy, evidences from the literature catalogue that using a team of traditional pursuit LA yields slow convergence. For instance, Tilak et al. [36] report thata team of traditional pursuit LA larger than 10 deployed for solving a deterministic combinatorial problem, namely sensor coverage, yields a very slow convergence speed. In fact, Tilak et al. state: "*Even at a modest number of 10 cameras, the centralized pursuit algorithm takes a long time for the automata team to converge which makes it unsuitable for an application like distributed object tracking where fast convergence is necessary*". Furthermore, some of the authors of the current manuscript [37] have also noticed this slow convergence in solving a machine learning classification problem mappedinto a combinatorial problem using a team of LA. Another important disadvantage of traditional teams of pursuit LA concerns the size of the required memory for storing the reward estimate vector. In the case of classical team of pursuit LA, one needs a shared memory for the reward estimate vector that increases dramatically with the size of the team. For instance, for a team of N LA each with two actions (binary action LA), the memory space required for storing the reward probability estimate is 2^N which is exhaustive as N increases [36]. This slow convergence of classical team of pursuit LA for solving deterministic optimization problems calls for a new LA paradigm which is the objective of this article.

In this paper, we develop a novel pursuit LA, which can be seen as the counterpart of the family of pursuit LA designed for stochastic environments [3]. While classical pursuit LAs are able to pursue the action with the highest reward estimate, our pursuit LA rather pursues the collection of actions that yield the highest performance. The theoretical analysis of the pursuit scheme does not follow classical LA proofs and can pave the way towards more schemes where LA can be applied to solve deterministic optimization problems.

We catalogue the contributions of this article as follows:

- We devise a simple and lightweight optimization framework based on the theory of LA. In contrast to any LA scheme presented in the literature, our solution is especially designed for deterministic environments.
- Our current solution extends the family of pursuit LA algorithms [3, 38, 39] to solve deterministic optimization problems. A common feature for all legacy pursuit algorithms is to estimate the reward probability of each action and pursue the action with the highest reward. In our current work, the environment is rather deterministic. Therefore, we opt to pursue the joint action of the team LA corresponding to the best solution found so far.
- We provide sound theoretical results that demonstrate the convergence of our scheme under both constant learning parameter and time-decaying learning parameter. To the best of our knowledge, this is the first work that proposes an analysis of LA scheme with time-decaying learning parameter.
- As an example of an optimization problem, we show how our scheme can be applied to solve the Max-SAT problem.

The remainder of this paper is organized as follows. In Section 2, we give an introduction to the theory of Learning Automata which is the fundamental tool in this paper. In Section 3, we survey some related work within the field of LA and optimization. In Section 4, we present our solution called Pursuit-LA and provide theoretical proofs demonstrating its convergence. Furthermore, we provide an experiment where we apply Pursuit-LA to the Max-SAT problem. Section 6 concludes the article.

2 Learning automata

In the field of Automata Theory, an automaton [5–7, 9, 10] is defined as a quintuple composed of a set of states, a set of outputs or actions, an input, a function that maps the current state and input to the next state, and a function that maps a current state (and input) into the current output.

Definition 1: A LA is defined by a quintuple <A, B, Q, $F(.,.)$, $G(.)$>, where:

1. $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of outputs or actions that the LA must choose from, and $\alpha(t)$ is the action chosen by the automaton at any instant t.
2. $B = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of inputs to the automaton. $\beta(t)$ is the input at any instant t. The set B can be finite or infinite. The most common LA input is $B = \{0, 1\}$, where $\beta = 0$ represents reward, and $\beta = 1$ represents penalty.
3. $Q = \{q_1, q_2, \ldots, q_s\}$ is the set of finite states, where $Q(t)$ denotes the state of the automaton at any instant t.
4. $F(.,.) : Q \times B \mapsto Q$ is a mapping in terms of the state and input at the instant t, such that, $q(t+1) = F[q(t), \beta(t)]$. It is called a *transition function*, i.e., a function that determines the state of the automaton at any subsequent time instant $t+1$. This mapping can either be deterministic or stochastic.
5. $G(.)$: is a mapping $G : Q \mapsto A$, and is called the *output function*. G determines the action taken by the automaton if it is in a given state as: $\alpha(t) = G[q(t)]$. With no loss of generality, G is deterministic.

If the sets Q, B and A are all finite, the automaton is said to be *finite*.

The Environment, E, typically, refers to the medium in which the automaton functions. The Environment possesses all the external factors that affect the actions of the automaton. Mathematically, an Environment can be abstracted by a triple <A, C, B>. A, C, and B are defined as follows:

1. $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of actions.
2. $B = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the is the output set of the Environment. Again, we consider the case when $m = 2$, i.e., with $\beta = 0$ representing a "Reward", and $\beta = 1$ representing a "Penalty".
3. $C = \{c_1, c_2, \ldots, c_r\}$ is a set of penalty probabilities, where element $c_i \in C$ corresponds to an input action α_i.

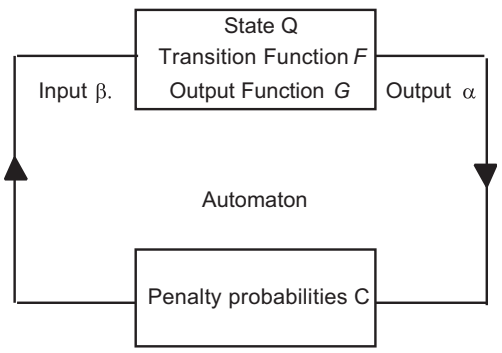

Fig. 1 Feedback Loop of LA

The process of learning is based on a learning loop involving the two entities: the Random Environment (RE), and the LA, as described in Fig. 1. In the learning process, the LA continuously interacts with the Environment to process responses to its various actions (i.e., its choices). Finally, through sufficient interactions, the LA attempts to learn the optimal action offered by the RE. The actual process of learning is represented as a set of interactions between the RE and the LA.

The automaton is offered a set of actions, and it is constrained to choosing one of them. When an action is chosen, the Environment gives out a response $\beta(t)$ at a time "t". The automaton is either penalized or rewarded with an unknown probability c_i or $1 - c_i$, respectively. On the basis of the response $\beta(t)$, the state of the automaton $\phi(t)$ is updated and a new action is chosen at (t + 1). The penalty probability c_i satisfies:

$$c_i = Pr[\beta(t) = 1 | \alpha(t) = \alpha_i] (i = 1, 2, \ldots, R).$$

We now provide a few important definitions used in the field. $P(t)$ is referred to as the action probability vector, where, $P(t) = [p_1(t), p_2(t), \ldots, p_r(t)]^T$, in which each element of the vector.

$$p_i(t) = Pr[\alpha(t) = \alpha_i], i = 1, \ldots, r, \text{ such that } \sum_{i=1}^{r} p_i(t) = 1 \ \ \forall t. \tag{1}$$

Given an action probability vector, $P(t)$ at time t, the *average penalty* is:

$$\begin{aligned} M(t) &= E[\beta(t)|P(t)] = Pr[\beta(t) = 1|P(t)] \\ &= \sum_{i=1}^{r} Pr[\beta(t) = 1|\alpha(t) = \alpha_i] \, Pr[\alpha(t) = \alpha_i] \\ &= \sum_{i=1}^{r} c_i p_i(t) \end{aligned}. \tag{2}$$

The average penalty for the "pure-chance" automaton is given by:

$$M_0 = \frac{1}{r} \sum_{i=1}^{r} c_i. \tag{3}$$

As $t \mapsto \infty$, if the average penalty $M(t) < M_0$, at least asymptotically, the automaton is generally considered to be better than the pure-chance automaton. $E[M(t)]$ is given by:

$$E[M(t)] = E\{E[\beta(t)|P(t)]\} = E[\beta(t)]. \tag{4}$$

A LA that performs better than by pure-chance is said to be *expedient*.

Definition 2: A LA is considered *expedient* if:

$$\lim_{t \to \infty} E[M(t)] < M_0.$$

Definition 3: A LA is said to be *absolutely expedient* if

$$E[M(t+1)|P(t)] < M(t),$$

implying that $E[M(t+1)] < E[M(t)]$.

Definition 4: A LA is considered *optimal* if

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

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

It should be noted that no optimal LA exist. Marginally sub-optimal performance, also termed above as ϵ-optimal performance, is what LA researchers attempt to attain.

Definition 5: A LA is considered ϵ-*optimal* if:

$$\lim_{n\to\infty} E[M(t)] < c_l + \epsilon, \tag{5}$$

where $\epsilon > 0$, and can be arbitrarily small, by a suitable choice of some parameter of the LA.

2.1 Types of learning automata

2.1.1 Deterministic learning automata

An automaton is termed as a *deterministic automaton*, if both the transition function $F(.,.)$ and the output function $G(.)$ are deterministic. Thus, in a deterministic automaton, the subsequent state and action can be uniquely specified, provided the present state and input are given.

2.1.2 Stochastic learning automata

If, however, either the transition function $F(.,.)$, or the output function $G(.)$ are stochastic, the automaton is termed to be a *stochastic automaton*. In such an automaton, if the current state and input are specified, the subsequent states and actions cannot be specified uniquely. In such a case, $F(.,.)$ only provides the probabilities of reaching the various states from a given state.

In the first LA designs, both the transition and output functions where time-invariant, and for this reason, these LA were considered to be "Fixed Structure Stochastic Automata" (FSSA). Tsetlin, Krylov, and Krinsky [12] have presented notable examples of this type of automata.

Subsequently, Vorontsova and Varshavskii introduced a class of stochastic automata known in the literature as Variable Structure Stochastic Automata (VSSA). In the definition of a VSSA, the LA is wholly defined by a set of actions (one of which is the output of the automaton), a set of inputs (which is usually the responsibility of the Environment) and a learning algorithm, T. The learning algorithm [7] operates on a vector (called *the Action Probability vector*).

Note that the algorithm T: $[0, 1]^R \times A \times B \to [0, 1]^R$ is an updating scheme where $A = \{\alpha_1, \alpha_2, \ldots, \alpha_R\}$, $2 \le R < \infty$, is the set of output actions of the automaton, and B is the set of responses from the Environment. Thus, the updating is such that.

$P(t+1) = T(P(t), \alpha(t), \beta(t))$, where P(t) is the action probability vector, $\alpha(t)$ is the action chosen at time t, and $\beta(t)$ is the response it has obtained.

If the mapping T is chosen in such a manner that the Markov process has absorbing states, the algorithm is referred to as an absorbing algorithm. Many families of VSSA that posses absorbing barriers have been reported [7]. Ergodic VSSA has also been investigated [7, 40]. These VSSAs converge in distribution and thus, the asymptotic distribution of the action probability vector has a value that is independent of the corresponding initial vector. While ergodic VSSA are suitable for non-stationary environments, absorbing VSSA are preferred in stationary environments.

3 Related work

In order to put our work in the right perspective, we will briefly discuss different optimization schemes relevant to this work mostly from the field of LA.

3.1 LA for optimization

A similar work to ours is due to Thathachar and Sastry [41] where the authors use a team of LA in order to find the optimal discriminant function in a feature space. The discriminant functions are parametrized, and a parameter is attached to each that is to be learned.

Subsequently, Santharam et al. [42] proposed using continuous LA in order to deal with the disadvantages of discretization, thus allowing an infinite number of actions. For an excellent review on the application of LA to the field of Pattern Recognition, we refer the reader to [43]. In [44], Zahiri devised an LA based classifier that operates using hypercubes in a recursive manner. In [45], the authors have proposed LA optimization methods for multimodal functions. Through experimental settings, the performance of these algorithms were shown to outperform genetic algorithms. In [46], the authors propose genetic LA for optimizing functions. Similarly, the work [47] proposed genetic algorithms for classifiers.

Misra and Oommen pioneered the concept of LA on a graph using pursuit LA [13, 48, 49] for solving the stochastic shortest path problem. Li [50] used a type of S Learning Automata [51] to find the shortest path in a graph. Beigy and Meybodi [52] provided the first proof in the literature that demonstrates the convergence of distributed LA on a graph for a reward inaction LA.

Concerning applications of distributed LA on a graph in the field of computer communications, we refer the reader to the work of Torkestani and collaborators [53–55].

3.2 Stochastic local search algorithms (SLS)

Due to their combinatorial explosion nature, large and complex SAT problems are hard to solve using systematic algorithms. One way to overcome the combinatorial explosion is to give up completeness. Local search algorithms are techniques which use this strategy. Local search algorithms are based on what is perhaps the oldest optimization method trial and error. Typically, they start with an initial assignment of values to variables randomly or heuristically generated. Satisfiability can be formulated as an optimization problem in which the goal is to minimize the number of unsatisfied clauses. Thus, the optimum is obtained when the value of the objective function equals zero, which means that all clauses are satisfied. Finite Learning Automata has been proposed as a mechanism for enhancing meta-heuristics based Max-SAT solvers. The work conducted in [56] proposes an adaptive memory based local search algorithm that exploits various strategies in order to guide the search to achieve a suitable trade-off between intensification and diversification. Multilevel techniques [57, 58] have been applied to Max-SAT with considerable success. They progressively coarsen the problem, find an assignment, and then employ a meta-heuristic to refine the assignment on each of the coarsened problems in reverse order.

During each iteration, a new solution is selected from the neighborhood of the current one by performing a move. Choosing a good neighborhood and a method for searching it is usually guided by intuition because very little theory is available as a guide. Most SLS uses a 1-flip neighborhood relation for which two truth-value assignments are neighbors if they differ in the truth value of one variable. If the new solution provides a better value in light of the objective function, the new solution becomes the current one. The search terminates if no better neighbor solution can be found.

One of the most popular local search for solving SAT is GSAT [59]. GSAT begins with a randomly generated assignment of values to variables and then uses the steepest descent heuristic to find the new variable-value assignment which best decreases the numbers of unsatisfied clauses. After a fixed number of moves, the search is restarted from a new random assignment. The search continues until a solution is found or a fixed number of restart is performed. An extension of GSAT referred to as random-walk [60] has been realized with the purpose of escaping from local optima. In a random walk step, a randomly unsatisfied clause is selected. Then, one of the variables appearing in that clause is flipped, thus effectively forcing the selected clause to become satisfied. The main idea is to decide at each search step whether to perform a standard GSAT or a random-walk strategy with a probability called the walk probability. Another widely used variant of GSAT is the WalkSAT algorithm originally introduced in [61]. It first picks randomly an unsatisfied clause c and then in a second step, one of the variables with the lowest break count appearing in the selected clause is randomly selected. The break count of a variable is defined as the number of clauses that would be unsatisfied by flipping the chosen variable. If there exists a variable with break count equals to zero, this variable is flipped, otherwise, the variable with minimal break count is selected with a certain probability (noise probability). The choice of unsatisfied clauses combined with the randomness in the selection of variables enables WalkSAT to avoid local minima and to explore the search space better. New algorithms [62] [63–65] have emerged using history-based variable selection strategy in order to avoid flipping the same variable. Apart from GSAT and its variants, several clause weighting based SLS algorithms [66, 67] have been proposed to solve SAT problems. The key idea is to associate the clauses of the given CNF formula with weights. Although these clause weighting SLS algorithms differ in the manner clause weights should be updated (probabilistic or deterministic) they all choose to increase the weights of all the unsatisfied clauses as soon as a local minimum is encountered. Clause weighting acts as a diversification mechanism rather than a way of escaping local minima. Finally, many other SLS algorithms have been applied to the SAT. These include techniques such as Simulated Annealing [68, 69], Evolutionary Algorithms [70], and Greedy Randomized Adaptive Search Procedures [71]. The nature-inspired GASAT algorithm [72] is a hybrid algorithm that combines a specific crossover and a tabu search procedure. The work in [73] proposes a hybrid approach called Iterated Robust Tabu Search (IRoTS) which combines an iterated local search and tabu search.

4 Our solution: Pursuit-LA

In this Section, we shall present our solution reckoned as Pursuit-LA for solving deterministic optimization problems. In many combinatorial problems, a candidate solution can be represented using a binary vector [74]. Adopting Pursuit-LA implies to attach an LA to each element of the binary vector whose respective decision is the action 0 or 1. The collective decision of the different LA will result into a solution. The solution with highest "fitness" will be pursued by the LA using the *LRI* scheme [7, 10]. Furthermore, we will give an example of application of the Pursuit-LA to the Max-SAT problem.

4.1 Convergence results of the pursuit-LA

In this Section, we will consider two convergence cases of the Pursuit-LA, namely convergence under time decaying learning parameter and convergence under constant learning parameter.

4.1.1 Pursuit-LA with time-dependent parameter

At each epoch, each LA in the team of LA chooses an action, therefore the choices of the team are synchronous. The joint action of the team of LA results in a candidate solution. The observed performance is fed back to the team of LA and used to reinforce the choice of the candidate solution yielding highest performance. More precisely for each LA in the team, we attach a component of the binary vector forming the candidate solution, the corresponding action coinciding with the candidate solution yielding highest performance sees its probability increasing at each time instant. In this sense, the joint action probability vector of the team of LA gets biased towards the best solution found so far, and thus the concept of pursuit. The choices of the team of LA are synchronous and the feedback is common for the team, which can be though as shared memory if one considers that the last feedback is stored in a common memory. Each LA also has a local memory to remember the best action so far (up to the current time instant) that has resulted in the highest performance for the team.

Let $C(t) = \{C_1(t), \ldots, C_m(t)\}$ be a candidate solution at time t where C_i takes a binary value and m the number of bits needed to code a candidate solution. We attach an LA to each component of the candidate solution.

The automaton's state probability vector at the component C_i at time t is $P_i(t) = [p_{(i,0)}(t), p_{(i,1)}(t)]$, which denotes the probability to yield 0 or 1 for the i^{th} component.

The normalized feedback function (or reward strength) is given by $f(C(t))$, where $C(t)$ is the candidate solution tested at instant t. The function $f(.)$ measures the fitness of the solution taking values from [0, 1] where 0 is the lowest possible reward, while 1 is the highest reward. In other words, the fitness function is normalized.

Let $C^*(t)$, be the solution with highest fitness found so far, i.e., the solution with highest fitness obtained up to time instant t.

The idea of pursuit here is to reward the LA whose actions correspond to the component of the solutions in $C^*(t)$.

We consider the LA update equations at component C_i. For all components C_i, and for, $j \in \{0, 1\}$, the update is given by:

$$p_{(i,j)}(t+1) = (1-\lambda_t)\delta_j + \lambda_t p_{(i,j)}(t) \tag{6}$$

Where δ_j is defined by

$$\delta_j = \begin{cases} 1 & \text{if } C_i^*(t) = j \\ 0 & \text{else} \end{cases} \tag{7}$$

λ_t is the update parameter and depends on time. In Theorem 13, we will consider the conditions by which the algorithm can converge when the update parameter depends on time. Further, we will give convergence results for the case of fixed λ, i.e., independent of time t.

Please note that, initially:

$$p_{(i,j)}(0) = \tfrac{1}{2}, \text{ for } j \in \{0, 1\}.$$

The informed reader would observe that the above update scheme corresponds to the linear Reward-Inaction LA update [1].

In fact if $j \notin C_i^*(t)$ then $p_{(i,j)}(t+1)$ is reduced by multiplying by λ_t which is less than 1 as per the following equation:

$$p_{(i,j)}(t+1) = \lambda_t p_{(i,j)}(t) \tag{8}$$

However if $j \in C_i^*(t)$ then $p_{(i,j)}(t+1)$ is increased. This can be proven as follows:

$$p_{(i,j)}(t+1) - p_{(i,j)}(t) = \left[(1-\lambda_t) + \lambda_t p_{(i,j)}(t)\right] - p_{(i,j)}(t) \tag{9}$$

$$= (1-\lambda_t) + p_{(i,j)}(t)(\lambda_t - 1) \tag{10}$$

$$= (1-\lambda_t)\left(1 - p_{(i,j)}(t)\right) \geq 0 \tag{11}$$

The update scheme is called pursuit LA and has rules that obey *LRI*. The idea is to always reward the transitions probabilities along the best solution obtained so far.

With the updating formula (Equation 6), we can show that the probability distribution converges to the distribution that satisfies the following property if the optimal solution C_i^* is unique.

$$p_{ij} = \begin{cases} 1 & \text{if } C_i^* = j \\ 0 & \text{else} \end{cases} \tag{12}$$

Intuition behind pursuit-LA Thathachar and Sastry [75] pioneered the idea of pursuit LA. The action with the highest reward estimate is "pursued". The latter work has fueled a great deal of interest in pursuit LA involving different variants [3, 38, 39]. A common feature for all these pursuit algorithms is to estimate the reward probability of each action and pursue the action with the highest reward. In our current work, the environment is ratherdeterministic. Therefore, we opt to pursue the joint action of the team LA, corresponding to the best solution found so far. We will now state some theoretical results that catalog the properties of the Pursuit-LA for both the time-varying update parameter and the fixed update parameter.

We will now state some theoretical results that catalogue the properties of the Pursuit-LA for both the time varying update parameter and the fixed update parameter. The optimal solution is generated with probability 1 only if the update parameter θ_t obeys the following condition:

$$\sum_{t=1}^{\infty} \prod_{k=1}^{t-1} \lambda_k = \infty \tag{13}$$

Proof.

The proof follows similar arguments as in [76]. Using recurrence, we can obtain a lower bound on $p_{(i,j)}(t)$:

$$p_{(i,j)}(t) \geq \prod_{k=1}^{t-1} \lambda_k p_{(i,j)}(0) \tag{14}$$

Let $p_{min}(0) > 0$ a lower bound on $p_{(i,j)}(0)$.

Let $A_t = \{C(t) \neq C^*\}$ the event that at iteration t, the candidate solution does not contain the optimal solution C^*.

Let B_T the event that optimal solution is not found up to instant T.

$$P(B_T) = \prod_{t=1}^{T} P(A_t) \tag{15}$$

$$P(B_T) \leq \prod_{t=1}^{T} \left(1 - \prod_{k=1}^{t-1} (\lambda_k p_{min}(0))^m\right) \tag{16}$$

By resorting to $(1-u) \leq exp\,(-u)$ we obtain

$$P(B_\infty) \leq \prod_{t=1}^{\infty} \left(1 - \prod_{k=1}^{t-1} (\lambda_k p_{min}(0))^m\right) \tag{17}$$

$$\leq \prod_{t=1}^{\infty} exp\left(- \prod_{k=1}^{t-1} (\lambda_k p_{min}(0))^m\right) \tag{18}$$

$$= exp\left(- \sum_{t=1}^{\infty} \prod_{k=1}^{t-1} \lambda_k^m p_{min}(0)^m\right) \tag{19}$$

However, from our assumption

$$\sum_{t=1}^{\infty} \prod_{k=1}^{t-1} \lambda_k = \infty$$

Since we have

$$P(B_\infty) \leq 0 \tag{20}$$

then

$$P(B_\infty) = P\left(C^* \text{never obtained}\right) = 0 \tag{21}$$

Examples of smoothing sequences which eventually generate the optimal solution with probability 1 (that is, which satisfy the sufficient condition of Theorem 1) includes.

$\lambda_t = 1 - 1/(t+1)^\beta$ for $\beta > 1$.

and

$\lambda_t = 1 - \frac{1}{(t+1)log(t+1)^\beta}$ for $\beta > 1$.

Let t^* the first time instant when the optimal solution is found, the optimal components are always reinforced. For $t^* + r$, for (i^*, j) such that $i^* \in C^*$ and $j \notin C^*$, we have using recurrence:

$$p_{(i^*,j)}\left(t^* + r\right) = \prod_{k=t^*}^{t^*+r-1} \lambda_k p_{(i^*,j^*)}(t) \tag{22}$$

Easy to see from the assumption that $\prod_{k=0}^{\infty} \lambda_k = 0$, by considering the log of the expression described in assumption on λ_k.

$$\lim_{r \to \infty} p_{(i^*,j)}\left(t^* + r\right) = 0 \tag{23}$$

By considering summation to 1 of probability of going from node i^*, and for j^* belonging to the optimal path.

$$\lim_{r \to \infty} p_{(i^*,j^*)}\left(t^* + r\right) = 1 \tag{24}$$

4.2 Constant update parameter

In Theorem 13, we give the convergence result of the Pursuit-LA for the case of fixed parameter λ that is independent of time.

The optimal solution is generated with probability 1 only if the update parameter $\lambda \to 1$.

Proof.

Using recurrence, we know that:

$$p_{(i,j)}(t) > \lambda^{t-1} p_{(i,j)}(0) \tag{25}$$

Thus,

$$P(A_t) \leq 1 - \left(\lambda^{t-1} p_{min}(0)\right)^m \tag{26}$$

Therefore,

$$P(B_T) \leq \prod_{t=1}^{T} \left(1 - \left(\lambda^{t-1} p_{min}(0)\right)^m\right) \tag{27}$$

Thus, by resorting again to $(1-u) \leq exp\,(-u)$ we obtain

$$P\left(C^* \text{never obtained}\right) = Prob(B_\infty) \leq \prod_{t=1}^{\infty} exp\left(-\lambda^{(t-1)m} p_{min}(0)^m\right) \tag{28}$$

$$\leq exp\left(-p_{min}(0)^m \sum_{t=1}^{\infty} \lambda^{(t-1)m}\right) \tag{29}$$

$$= exp\left(-p_{min}(0)^m \sum_{t=0}^{\infty} \lambda^{tm}\right) \tag{30}$$

Let us define $h(\alpha) = \sum_{t=0}^{\infty} \lambda^{tm}$.

$$h(\alpha) = \sum_{t=0}^{\infty} \lambda^{tm} \tag{31}$$

$$= 1/(1-\lambda^m) \tag{32}$$

$$\lim_{T\to\infty} P(B_T) = P\left(C^* \text{ never obtained}\right) \tag{33}$$

$$\leq exp(-p_{min}(0)^m h(\lambda)) \tag{34}$$

Since we know that $\lim_{\lambda\to 1} h(\lambda) = \infty$, then $\lim_{T\to\infty} P(B_T)$ can be made arbitrarily close to zero, if λ approaches 1.

Hence the theorem is proven. Now, let us characterize the LA probabilities at convergence.

Let t^* the first time instant when the optimal solution is found, the optimal so far components are always reinforced. For $t^* + r$, for $(i^*, j^*) \in C^*$, we have:

Using recurrence, we can obtain verify

$$p_{(i^*,j^*)}(t+r) = \lambda^r p_{(i^*,j^*)}(t) + (1-\lambda)\sum_{i=0}^{r-1} \lambda^{r-i-1} \tag{35}$$

We remark

$$\lim_{r\to\infty} \sum_{i=0}^{r-1} \lambda^{r-i-1} = 1/(1-\lambda) \tag{36}$$

Therefore,

$$\lim_{r\to\infty} p_{(i^*,j^*)}(t+r) = (1-\lambda) \times 1/(1-\lambda) = 1 \tag{37}$$

4.3 Pursuit-LA with artificial barriers

In this section, we extend our Pursuit-LA with the concept of artificial barriers introduced recently by Yazidi and Hammer [4] to avoid the lock-in probability effect. The presented Pursuit-LA in the previous section is an absorbing scheme where the team of LA will converge after a large number of iterations to an absorbing state composed of a vector with components either 0 or 1. This creates a challenge when it comes to tuning the learning parameter as choosing high values of the learning parameter close to 1 renders the schemes extremely slow, while choosing high values might lead to premature convergence. To allow the scheme to avoid still getting locked in an absorbing state where premature convergence can take place, we introduce an upper and lower band for the probability of each LA in the team. Therefore, instead of allowing the LA probabilities to admit values within the interval [0, 1], we force the probabilities to be located in $[p_{min}, p_{max}]$ where p_{max} is a user-defined upper bound for all the $p_{(i,j)}(t)$ and $p_{min} = 1 - p_{max}$ the counter part lower bound. p_{max} needs to be chosen in the neighborhood of 1 in order to bias the exploration to the neighborhood of the best solution found so far. We shall give now the update equations for Pursuit-LA with artificial barriers. For all components C_i, and for, $j \in \{0, 1\}$, the update is given by:

$$p_{(i,j)}(t+1) = (1-\lambda)\left(\delta_j p_{max} + \left(1-\delta_j\right)p_{min}\right) + \lambda p_{(i,j)}(t) \tag{38}$$

Where δ_j is defined by

$$\delta_j = \begin{cases} 1 & \text{if } C_i^*(t) = j \\ 0 & \text{else} \end{cases} \tag{39}$$

Please note that, if we initially impose that $p_{max} \leq p_{(i,j)}(0) \leq p_{min}$, then it is easy to prove by recurrence that the update form will guarantee that at any subsequent time $t > 0$ that $p_{max} \leq p_{(i,j)}(t) \leq p_{min}$.

Let us suppose that $p_{max} \leq p_{(i,j)}(t) \leq p_{min}$ and prove $p_{max} \leq p_{(i,j)}(t+1) \leq p_{min}$ In fact, $p_{(i,j)}(t+1)$ can be written in the form $p_{(i,j)}(t+1) = (1-\lambda)p + \lambda p_{(i,j)}(t)$ where $p = p_{max}$ or $p = p_{min}$ depending on whether $\delta_j = 1$ or $\delta_j = 0$. Therefore, $p_{(i,j)}(t+1)$ is a convex combination of two quantities that bother are in the interval $[p_{min}, p_{max}]$. Hence, the result is proven by recurrence.

It is easy to see that the update equation (Eq. (38)) can be written differently. In fact, if $j \notin C_i^*(t)$ then $p_{(i,j)}(t+1)$ reduces to:

$$p_{(i,j)}(t+1) = (1-\lambda)p_{min} + \lambda p_{(i,j)}(t) \tag{40}$$

However, if $j \in C_i^*(t)$ then $p_{(i,j)}(t+1)$ reduces to:

$$p_{(i,j)}(t+1) = (1-\lambda)p_{max} + \lambda p_{(i,j)}(t) \tag{41}$$

We can show that if $j \in C_i^*(t)$ then $p_{(i,j)}(t+1)$ increases while its decreases in the opposite case (i.e, $j \in C_i^*(t)$). In fact,

$$p_{(i,j)}(t+1) - p_{(i,j)}(t) = (1-\lambda)\left(\delta_j p_{max} + \left(1-\delta_j\right)p_{min}\right) + \lambda p_{(i,j)}(t) - p_{(i,j)}(t) \tag{42}$$

$$= (1-\lambda_t)\left(\delta_j p_{max} + \left(1-\delta_j\right)p_{min}\right) - p_{(i,j)}(t) \tag{43}$$

Then if $j \in C_i^*(t)$

$$p_{(i,j)}(t+1) - p_{(i,j)}(t) = (1-\lambda_t)\left(p_{max} - p_{(i,j)}(t)\right) \geq 0 \tag{44}$$

Whereas if $j \notin C_i^*(t)$

$$p_{(i,j)}(t+1) - p_{(i,j)}(t) = (1-\lambda_t)\left(p_{min} - p_{(i,j)}(t)\right) \leq 0 \tag{45}$$

It is easy to observe that whenever $p_{max} = 1$, and consequently $p_{min} = 0$, the Pursuit-LA with absorbing barriers reduces to the Pursuit-LA with fixed learning parameter introduced in the previous section.

Before closing this section, it is not of place to observe that our algorithm enjoys low computational complexity. In fact, the proposed Pursuit-LA requires only an order of m operations per time step. This low computational complexity is an inherent property of Reinforcement Learning algorithms in general and LA in particular.

5 Application of pursuit-LA to max-SAT problem

In this section, we will test the performance of our algorithm for solving the Max-SAT problem, which is a well-known class of deterministic optimization problems. We will examine two main aspects of the algorithm. The first aspect is investigated in Section 1 and concerns the sensitivity of the algorithm to changes in the learning parameter. The second aspect we tackle is how well the current algorithm compares to other well-established state-of-the-art MAX-SAT solvers. The latter aspect is treated in Section 2.

5.1 Effect of varying the learning parameter

We will use a benchmark from https://www.cs.ubc.ca/hoos/SATLIB/benchm.html related to the SAT-encoded Flat Graph Colouring Problems. As a criterion of convergence, we reckon that the algorithm has converged whenever each of the LA has converged, meaning that each LA has an action whose probability exceeds $1 - \epsilon$ where ϵ denotes a small scalar. In all the experiments,we choose $\epsilon = 0.01$. In more formal terms, we deem that the scheme has converged if for all i, $p_{(i,0)}(t) > 1 - \epsilon$ or $p_{(i,1)}(t) > 1 - \epsilon$. As an objective function, we resort to the percentage of satisfied clauses which also characterizes the performance of the algorithm. In the Max-SAT problem, we would like to maximize the number of satisfied clauses.

Table 1 gives the performance of the Pursuit-LA and convergence time for three values of the learning parameters $\lambda = 0.9$, $\lambda = 0.99$ and $\lambda = 0.999$. PC denotes the percentage of satisfied clauses while CT denotes the convergencetime. In Table 1, we consider different files representing the SAT-encoded Flat Graph Colouring Problems. The first category of files with prefix flat30 denotes a problem with 100 instances, 30 vertices, 60 edges, 3 colours, 90 variables and 300 clauses. The second category of files with prefix flat200 denotes a problem with 100 instances, 200 vertices, 479 edges, 3 colours, 600 variables and 223 clauses.

Table 1 Performance of the Pursuit-LA and convergence time for different values of the learning parameter

	$\lambda = 0.9$		$\lambda = 0.99$		$\lambda = 0.999$	
	PC	CT	PC	CT	PC	CT
flat200–1	0.8341	169	0.8971	1710	0.9535	18,899
flat200–2	0.8341	116	0.8922	1483	0.9544	19,958
flat200–3	0.8399	147	0.8909	1626	0.9530	17,946
flat200–4	0.8332	146	0.9105	1739	0.9427	21,997
flat200–5	0.8377	145	0.9016	1776	0.9552	20,102
flat30–1	0.9	88	0.9666	1170	0.9833	11,100
flat30–2	0.9	108	0.9633	1123	0.9833	10,656
flat30–3	0.9133	98	0.9566	1306	0.98	9815
flat30–4	0.91	116	0.9666	1061	0.98	10,961
flat30–5	0.9233	107	0.9633	1016	0.99	11,223
flat30–6	0.8933	90	0.9633	1113	0.98	10,475

From Table 1, the general remark is that the Pursuit-LA algorithm is generally fast and yields acceptable performance even for low values of λ. The performance seems to increase as λ increases; however, at the cost of longer convergence time. For example, let us consider the file flat30–5. We observe that Pursuit-LA converges quite fast for a small learning parameter $\lambda = 0.99$, namely with 107 iterations. Whenever we increase the parameter to $\lambda = 0.999$, the convergence time increases considerably to 11,223; however, the performance increases too substantially from 0.9233 to 0.99. In Fig. 2, we also give the evolution of the performance of the algorithm using the same file flat30–5 for $\lambda = 0.999$. Initially, i.e., at time zero, the performance was 0, 7533. As time proceeds, we observe from Fig. 2 that the performance steadily increases until converging to 0.99 after around 11,223 iterations. Please note that the first time we reach this performance is after about 8000 iterations. Nevertheless, it takes more iterations for the Pursuit-LA to converge as the probabilities of the actions that yield this performance (0.99) will keep on increasing as long no other better solution. This increase in the probability takes place with a learning parameter as big as $\lambda = 0.999$. The remaining time to converge after finding the best solution so far is depending on the time it takes for the smallest action probability to have its probability increasing above $1 - \epsilon$.

5.2 Comparison against reference solvers

The benchmark instances which are used to evaluate the performance of the Pursuit-LA algorithm belong to Random Unweighted-MAX2SAT/MAX3SAT.[1]

The performance of the algorithm is compared to various popular solvers in the literature:

[1] Those instances can be found in http://infohost.nmt.edu/borchers/maxsat.html.

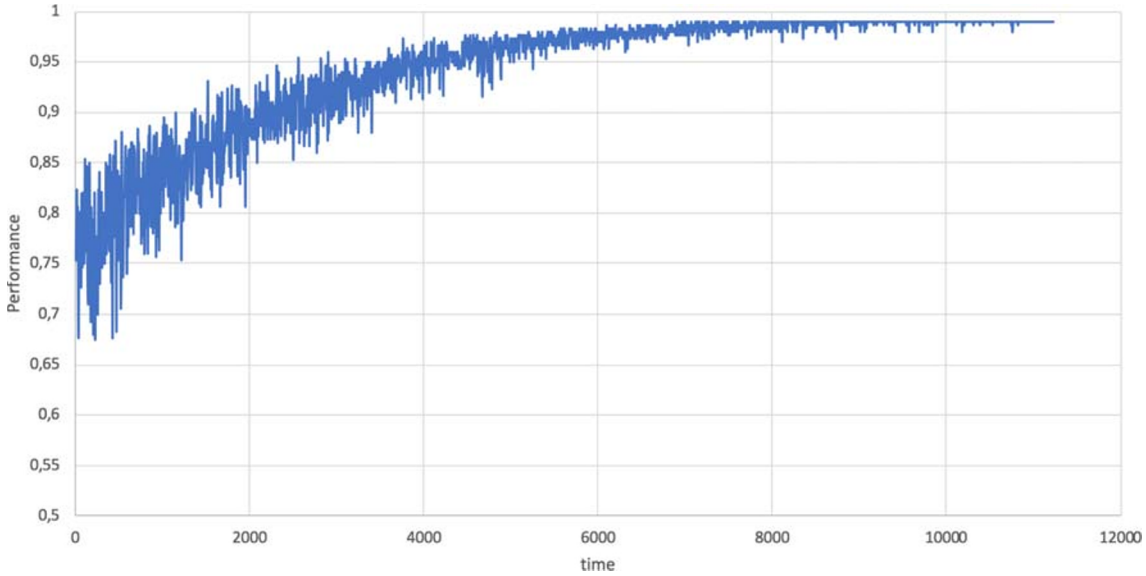

Fig. 2 Example of performance over time of the Pursuit-LA for flat30–5, with $\lambda = 0.999$

- AdaptNovelty+: stochastic local search algorithm with an adaptive noise mechanism [77].
- IRoTS: Iterated Robust Tabu Search algorithm [73].
- RoTS: Robust Tabu Search algorithm [78].
- AdaptG2WSATp: adaptive gradient-based greedy WalkSAT algorithm with promising decreasing variable heuristic [79]
- Adaptive memory-based local search heuristic (denoted by AMLS1,AMSL2) [56].

For the reference algorithms (IRoTS, RoTS, AdaptNovelty+) we carry out the experiments using UBCSAT (version 1.1) an implementation and experimentation environment for stochastic local search algorithms for SAT and MAX-SAT solvers. As shown by Tompkins and Hoos [80], the implementation of these reference algorithms in UBCSAT is more efficient than (or just as efficient as) the original implementations. In all tables, the first and second column identify the problem instance and the best known objective value $f*$ (number of unsatisfied clauses). The remaining columns give the results of the the algorithms using three performance quality criteria which have been widely used for the performance evaluation of stochastic local search [56]:

- The average objective function f_{av} over 20 independent runs.
- The success rate sr defined as the number of time the algorithm reaches the best known objective value over 20 runs.
- The average search step for reaching the average value.

To avoid premature convergence, we use $p_{max} = 0.9$. Furthermore, we fix $\lambda = 0.95$. We run the algorithms in epochs each consisting of 1000 iterations. We only update the probabilities at the end of each epoch. In simple terms, we test the current probability vector during the whole epoch before updating it best on the best solution found so far in the entire epoch.

Tables 2, 3 and 4 give the comparisons results.

To summarize the results of this section, we have incorporated comparisons against some of the most established state-of-the-art schemes including AMSL1, AMSL2, AdaptiveNovelty+, AdaptG2WSATp, RoTS IroTS using different benchmarks. The results are really conclusive and quite surprising too. Although our scheme is straightforward and we aimed to show that this a proof of concept of the possibility to use LA for solving deterministic problems, we have found that it performs well. It is even superior to the AdaptNovelty+ which is a sophisticated state-of-art Max-SAT algorithm. For example, in Table 4, our Pursuit-LA consistently outperforms the AdaptiveNovelty+ in terms of f_{av} and convergence steps. Observe, for example, the results for the file s2v120c1700–2. Our Pursuit-LA achieves several unsatisfied clauses 250 in 306 epochs while the AdaptiveNovelty+ achieves 261 in 25,229 iterations. The optimal number for this case is 248, and therefore our Pursuit-LA has just two unsatisfied clauses compared to the AdaptiveNovelty+ that has eleven unsatisfied clauses.

5.3 Discussion

We believe that the Pursuit-LA algorithm owes it's performance to two main design principles:

- By adding some artificial barriers, p_{min} and p_{max} for the LA probability, we are able to achieve better results by

Table 2 Performance Comparisons: Unweighted MAX2SAT/MAX3SAT

	f*	Pursuit-LA			AMSL2			AdaptiveNovelty+			AdaptG2WSATp		
Inst		f_{av}	sr	*steps*	f_{av}	sr	*steps*	f_{av}	sr	*steps*	f_{av}	sr	*steps*
p2200/100	5	5	20	168	5	20	417	5	20	478	5	20	150
p2300/100	15	15	19	416	15	20	152	15	20	817	15	20	175
p2400/100	29	31	8	135	29	20	2006	29.5	10	354,426	29	16	43,585
p2500/100	44	44	20	168	44	20	292	45	5	358,968	44	20	8885
p2600/100	65	66	17	166	65	20	299	67	1	387,088	65	20	21,354
p3500/100	4	10	1	316	4	20	14,771	4	20	2646	4	20	1952
p3550/100	5	9	1	164	5	20	7784	5	20	3795	5	20	1561
p3600/100	8	11	1	776	8	20	4829	8	20	8299	8	20	3861
p2300/150	4	4	20	250	4	20	210	4	20	237	4	20	216
p2450/150	22	23	18	224	22	20	398	22	20	35,985	22	20	3917
p2600/150	38	39	17	478	38	20	15,283	39	0	–	38	2	42,976
p3675/150	2	7	3	949	2	20	5674	2	20	22,831	2	20	6705
p3750/150	5	10	1	322	5	20	11,906	5	20	8137	5	20	5237

avoiding premature convergence of the algorithm. In fact, during the first experiments we had without artificial barriers, we observed some low performance due to the so-called stagnation effect. Stagnation means that the best so far solution does not change for a certain number of iterations that is relatively long. In face of stagnation and in the absence of artificial barriers, the LA team will get trapped into a probability vector which corresponds to an exclusive choice of the best so far solution. This happens within finite number of iterations that depends onthe learning parameter.

- Furthermore, the second appealing idea that makes the algorithm performant is pursuing the best solution found so far in the probability space. In this perspective, pursuing means, at each iteration, biasing gradually the probabilistic search towards the optimal solution found so far in a probabilistic manner. In fact, the role of the learning parameter is to adjust the quantity by which the change in probability takes place in the direction of the best solution found so far.

Despite that the performance of Pursuit-LA is promising, it does not outperform the other algorithms in all scenarios, namely RoTS and IRoTS are superior to Pursuit-LA in terms of the quality of the solution as seen in Table 4. In fact, althoughthe Pursuit-LA gives a satisfactory solutions it

Table 3 Performance Comparisons: Unweighted MAX2SAT/MAX3SAT

	f*	Pursuit-LA			AMSL1			IRoTS			erline RoTS		
Inst		f_{av}	sr	*steps*	f_{av}	sr	*steps*	f_{av}	sr	*steps*	f_{av}	sr	*steps*
p2200/100	5	5	20	168	5	20	222	5	20	672	5	20	920
p2300/100	15	15	19	416	15	20	220	15	20	127	15	20	200
p2400/100	29	31	8	135	29	20	3083	29	20	505	29	20	513
p2500/100	44	44	20	168	44	20	359	44	20	125	44	20	141
p2600/100	65	66	17	166	65	20	369	65	20	158	65	20	151
p3500/100	4	10	1	316	4	20	43,051	4	20	1939	4	20	2314
p3550/100	5	9	1	164	5	20	6272	5	20	2230	5	20	3775
p3600/100	8	11	1	776	8	20	4368	8	20	1817	8	20	2053
p2300/150	4	4	20	250	4	20	208	4	20	226	4	20	263
p2450/150	22	23	18	224	22	20	354	22	20	374	22	20	580
p2600/150	38	39	17	478	38	20	5188	38	20	1807	38	20	3564
p3675/150	2	7	3	949	2	20	42,153	2	20	5674	2	20	22,831
p3750/150	5	10	1	322	5	20	41,177	5	20	4737	5	20	9262

Table 4 Performance Comparison: Unweighted MAX2SAT/MAX3SAT

	f*	RoTS			IRoTS			AdaptNovelty+			erline Pursuit-LA		
Inst		f_{av}	sr	*steps*	f_{av}	sr	*steps*	f_{av}	sr	*steps*	ff_{av}	sr	*steps*
s2v120c1700–1	**257**	257	20	458	257	20	540	280	0	39,770	273	0	236
s2v120c1700–2	**248**	248	20	1460	248	20	1097	261	0	25,229	250	12	306
s2v120c1700–3	**239**	239	20	538	239	20	1729	253	0	39,321	240	17	336
s2v120c1800–1	**291**	291	20	559	291	20	630	306	0	42,222	299	2	326
s2v120c1800–3	**279**	279	20	330	279	20	160	293	0	34,378	279	19	368
s2v120c2600–2	**458**	458	20	246	458	20	695	486	0	38,513	466	12	252
s2v120c2600–3	**440**	440	20	325	440	20	359	463	0	50,431	443	17	224
s2v140c1600–2	**221**	221	20	785	221	20	1190	237	0	66,145	229	2	290
s2v140c2000–2	**308**	308	20	396	308	20	357	330	0	46,133	309	19	267
s2v140c2600–1	**422**	422	20	556	422	20	1047	448	0	79,668	432	2	293
s2v140c2600–3	**406**	406	20	811	406	20	693	433	0	75,895	413	4	313
s2v200c1200–2	**127**	127	20	775	127	20	540	136	0	26,332	137	6	647
s2v200c1600–4	**195**	195	20	1285	195	20	1172	216	0	41,019	220	0	275
s3v70c700–5	**22**	22	20	125	22	20	111	22	20	1841	22	20	106
s3v70c900–9	**35**	35	20	497	35	20	601	35	20	5650	38	11	80
s3v70c1500–1	**90**	90	20	575	90	20	928	90	2	12,527	92	15	421
s3v80c600–3	**11**	11	20	499	11	20	1012	11	20	3366	13	10	165
s3v80c1000–1	**44**	44	20	1021	44	20	865	45	8	25,481	44	19	454
s3v90c1000–1	**34**	34	20	259	34	20	266	34	20	22,117	37	12	463
s3v90c1100–2	**44**	44	20	868	44	20	1233	44	3	16,705	44	20	269
s3v90c1200–5	**59**	59	20	1111	59	20	1134	60	0	18,959	62	7	169
s3v110c800–7	**16**	16	20	1063	16	20	1332	16	20	11,018	19	3	286
s3v110c900–2	**22**	22	20	1446	22	20	1697	22	20	16,325	27	0	174
s3v250c1000–1	**5**	5	14	13,236	6	4	27,552	6	13	68,690	26	0	479

converges prematurely according to the results of Table 4. For instance, for s2v200c1200–2, $f^* = 127$ is achieved for RoTS and IRoTS, while the Pursuit-LA achieves 137 unsatisfied clauses but within less time, namely, 647 epochs compared to 775 and 540 steps for RoTS and IRoTS respectively. Therefore, we believe that adaptively adjusting the learning parameter as well as the barriers over time borrowing ideas from IRoTS will boost the performance. Under artificial barriers, our stopping criterion used in the simulations in Section 2 is the stagnation of the search for two consecutive epochs. One can deal with stagnation using different ideas in the literature. For instance, IRoTS forces any variable whose value has not been changed over the last 10 search steps to be flipped. Such enhancement to Pursuit-LA can be the object of future research.

The idea behind Pursuit-LA is to bias the search probability vector towards the optimal solution found so far. However, adequately tuning the the learning parameter is a challenge and can be the objective of further future research. As shown in the experiments in Table 2, a small value of the tuning part will fasten the convergence speed at the cost of low quality solution. However, a large value of the tuning parameter induces a slow convergence speed, which usually results in a good quality solution. We have also improved the algorithm by imposing some artificial barriers. By virtue of the artificial barriers, we avoid the lock in probability which is a phenomenon known in the field of LA as each individual i^{th} LA will have its probability vector $P_i(t) = [p_{(i,0)}(t), p_{(i,1)}(t)]$ converging to [0, 1] or [1, 0] which stops the search. Artificial barriers can help to solve local optimum problem. However, our algorithm can be adjusted to allow more diversification of the solution by using similar procedures to genetic algorithms to diversify the solution for example through mutation and crossover operations.

The Pursuit-LA algorithm is a stochastic algorithm. It has been compared in Table 2 with 3 stochastic algorithms RoTS, IRoTS and AdaptNovelty+ using the software UBCSAT [80]. RoTS is an algorithm based on a tabu search that repeatedly chooses the value of the tabu tuning parameter at random from a given interval during the search. The variant IRoTS whose subsidiary local search phase and perturbation phase are both based on RoTS uses a randomized acceptance criterion that is biased towards better-quality candidate solutions. The noise parameter, p, which controls the degree of randomness of the

search process, has a major impact on the performance and run-time behaviour of the original algorithm Novelty+. Unfortunately,the optimal value of p varies significantly between problem instances, and even small deviations from the optimal value can lead to substantially decreased performance. AdaptNovelty+ dynamically adjusts the noise setting p based on search progress. It gave similar results in 4 cases when compared to both RoTS, IRoTS and was beaten in the remaining cases by at most 6%. However, our algorithm converges faster. The authors believe that they could improve the quality of solutionsgiven by algorithm by finding a suitable balance between diversification and intensification. By adopting a similar technique to AdaptNovelty+, we can increase the noise, by lowering down p_{max} and thus allowing more non-greedy moves, i.e., moves not in the neighborhood of the best solution so far. The strength of the algorithm is the fact that it could be parallelized and used in a multilevel context so that diversification and intensification could be exploited at different levels of the multilevel strategy. When Pursuit-LA algorithm is compared to these three algorithms, one needs to compare the strategy of adopting diversification and intensification between the different algorithms and which of these strategies is more efficient. The comparison may lead toa hybrid approach as the one described in [81].

Possible applications of pursuit-LA Within the class of deterministic optimization problems, there is a large family of combinatorial problems which are by definition NP-hard. Those problems are solved usually using algorithms such as genetic algorithms, tabu-search, simulated annealing, Ant-Colony Optimization (ACO) to mention a few. Examples of combinational problems that can solved by our proposed solution include traveling salesman problems, knapsack problems, job scheduling, graph coloring, quadratic assignment [82] etc... In the current paper, we have dealt with an optimization problem where the candidate solutions can be represented in a binary format. Nevertheless, it is straightforward to extend our solution to code non-binary solutions by using the concept of multi-action LA. In this sense, a candidate solution is coded as a vector of discrete variables and therefore a multi-action LA can be attached to each component of the vector representing the candidate solution. A promising research direction is also to solve deterministic continuous optimization problems using the concept of pursuit. In fact, the components of the m dimensional vectors can be drawn randomly by using a team of m individual CALA [83].

6 Conclusion

In this paper, we have provided a novel LA that can solve deterministic optimization problems based on the idea of pursuit. The search for an optimal solution is conducted using a team of cooperative LA. The scheme can be seen as a game-theoretical solution to a deterministic optimization problem. Apart from being a contribution to the field of LA in itself, the scheme is lightweight and extremely simple to implement with very little memory. Despite being appealingly simple, extensive experimental results demonstrate that it performs well compared to sophisticated state-of-the-art approaches. As future work, we aim to investigate further improving the performance of the pursuit scheme in terms of convergence speed and exploration of the search space using Multilevel techniques introduced by Bouhmala [57, 58].

Funding Information Open Access funding provided by OsloMet - Oslo Metropolitan University.

References

1. Agache M, Oommen BJ (2002) Generalized pursuit learning schemes: new families of continuous and discretized learning automata. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32(6):738–749
2. Thathachar MAL, Sastry PS (2002) Varieties of learning automata: an overview. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32(6):711–722
3. Agache M, Oommen BJ (2002) Generalized pursuit learning schemes: new families of continuous and discretized learning automata, IEEE transactions on systems, man, and cybernetics. Part B (Cybernetics) 32(6):738–749
4. Yazidi A, Hammer HL (2018) Solving stochastic nonlinear resource allocation problems using continuous learning automata. Appl Intell 48(11):4392–4411
5. Lakshmivarahan S (1981) Learning Algorithms Theory and Applications, Springer-Verlag
6. Najim K, Poznyak AS (1994) Learning automata: theory and applications. Pergamon Press, Oxford
7. Narendra KS, Thathachar MAL (1989) Learning automata: an introduction. Prentice-Hall, Inc.
8. Obaidat MS, Papadimitriou GI, Pomportsis AS (2002) Learning automata: theory, paradigms, and applications. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32(6):706–709
9. Poznyak AS, Najim K (1997) Learning automata and stochastic optimization. Springer-Verlag, Berlin
10. Thathachar MAL, Sastry PS (2003) Networks of learning automata: techniques for online stochastic optimization. Kluwer Academic, Boston

11. Zhang J, Wang C, Zang D, Zhou M (2016) Incorporation of optimal computing budget allocation for ordinal optimization into learning automata. IEEE Trans Autom Sci Eng 13(2):1008–1017
12. Tsetlin ML (1973) Automaton theory and the modeling of biological systems. Academic Press, New York
13. Misra S, Oommen BJ (2004) GPSPA: a new adaptive algorithm for maintaining shortest path routing trees in stochastic networks. Int J Commun Syst 17:963–984
14. Obaidat MS, Papadimitriou GI, Pomportsis AS, Laskaridis HS (2002) Learning automata-based bus arbitration for shared-edium ATM switches. IEEE Trans Syst Man Cybern B 32:815–820
15. Oommen BJ, Roberts TD (2000) Continuous learning automata solutions to the capacity assignment problem. IEEE Trans Comput C-49:608–620
16. Papadimitriou GI, Pomportsis AS (2000) Learning-automata-based TDMA protocols for broadcast communication systems with bursty traffic. IEEE Communication Letters:107–109
17. Atlassis AF, Loukas NH, Vasilakos AV (2000) The use of learning algorithms in ATM networks call admission control problem: a methodology. Comput Netw 34:341–353
18. Atlassis AF, Vasilakos AV (2002) The use of reinforcement learning algorithms in traffic control of high speed networks, Advances in Computational Intelligence and Learning 353–369
19. Vasilakos AV, Saltouros MP, Atlassis AF, Pedrycz W (2003) Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques, IEEE transactions on systems. Man and Cybernetics: Part C 33:297–312
20. Seredynski F (1998) Distributed scheduling using simple learning machines. Eur J Oper Res 107:401–413
21. Kabudian J, Meybodi MR, Homayounpour MM (2004) Applying continuous action reinforcement learning automata (CARLA) to global training of hidden markov models, in: Proceedings of the International Conference on Information Technology: Coding and Computing , ITCC'04, Las Vegas, Nevada, pp. 638–642
22. Meybodi MR, Beigy H (2002) New learning automata based algorithms for adaptation of backpropagation algorithm pararmeters. Int J Neural Syst 12:45–67
23. Unsal C, Kachroo P, Bay JS (1997) Simulation study of multiple intelligent vehicle control using stochastic learning automata. Transactions of the Society for Computer Simulation International 14:193–210
24. Oommen BJ, Croix E d S (1995) Graph partitioning using learning automata. IEEE Trans Comput C-45:195–208
25. Collins JJ, Chow CC, Imhoff TT (1995) Aperiodic stochastic resonance in excitable systems. Phys Rev E 52:R3321–R3324
26. Cook RL (1986) Stochastic sampling in computer graphics. ACM Trans Graph 5:51–72
27. Barzohar M, Cooper DB (1996) Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. IEEE Trans Pattern Anal Mach Intell 7:707–722
28. Brandeau ML, Chiu SS (1989) An overview of representative problems in location research. Manag Sci 35:645–674
29. C. Bettstetter, H. Hartenstein, Xavier Pérez-Costa, Stochastic properties of the random waypoint mobility model, Journal Wireless Networks 10 (2004) 555–567
30. B. S. Rowlingson, P. J. Diggle, SPLANCS: Spatial Point Pattern Analysis Code in S-Plus, University of Lancaster, North West Regional Research Laboratory, 1991
31. Paola M (1998) Digital simulation of wind field velocity. J Wind Eng Ind Aerodyn 74-76:91–109
32. Cusumano JP, Kimble BW (1995) A stochastic interrogation method for experimental measurements of global dynamics and basin evolution: application to a two-well oscillator. Nonlinear Dynamics 8:213–235
33. Baddeley A, Turner R (2005) Spatstat: an R package for analyzing spatial point patterns. J Stat Softw 12:1–42
34. Sastry P, Nagendra G, Manwani N (2010) A team of continuous-action learning automata for noise-tolerant learning of half-spaces. IEEE Transactions on Systems, Man, and Cybernetics 40(1):19–28
35. Granmo O, Oommen B, Myrer S, Olsen M (2007) Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37(1):166–175
36. Tilak O, Mukhopadhyay S, Tuceryan M, Raje R (2010) A novel reinforcement learning framework for sensor subset selection, in: 2010 International Conference on Networking, Sensing and Control (ICNSC), IEEE, pp. 95–100
37. M. Goodwin, A. Yazidi, T. M. Jonassen, Distributed learning automata-based s-learning scheme for classification, Pattern Analysis and Applications (2019) 1–16
38. Zhang X, Granmo O-C, Oommen BJ (2013) On incorporating the paradigms of discretization and bayesian estimation to create a new family of pursuit learning automata. Appl Intell 39(4):782–792
39. Oommen BJ, Lanctôt JK (1990) Discretized pursuit learning automata. IEEE Transactions on Systems, Man, and Cybernetics SMC-20(4):931–938
40. Oommen BJ, Agache M (2001) Continuous and discretized pursuit learning schemes: various algorithms and their comparison. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 31:277–287
41. Thathachar MA, Sastry PS (1987) Learning optimal discriminant functions through a cooperative game of automata. IEEE Transactions on Systems, Man and Cybernetics 17(1):73–85
42. Santharam G, Sastry P, Thathachar M (1994) Continuous action set learning automata for stochastic optimization. Journal of the Franklin Institute 331(5):607–628
43. Sastry P, Thathachar M (1999) Learning automata algorithms for pattern classification. Sadhana 24(4):261–292
44. Zahiri S (2008) Learning automata based classifier. Pattern Recogn Lett 29(1):40–48
45. Zeng X, Liu Z (2005) A learning automata based algorithm for optimization of continuous complex functions. Inf Sci 174(3): 165–175
46. Howell M, Gordon T, Brandao F (2002) Genetic learning automata for function optimization. IEEE Transactions on Systems, Man, and Cybernetics 32(6):804–815
47. Bandyopadhyay S, Murthy CA, Pal SK (1995) Pattern classification with genetic algorithms. Pattern Recogn Lett 16(8):801–808
48. Misra S, Oommen BJ (2005) Dynamic algorithms for the shortest path routing problem: learning automata-based solutions. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35(6):1179–1192
49. Misra S, Oommen BJ (2006) An efficient dynamic algorithm for maintaining all-pairs shortest paths in stochastic networks. IEEE Trans Comput 55(6):686–702
50. Li H, Mason L, Rabbat M (2009) Distributed adaptive diverse routing for voice-over-ip in service overlay networks. IEEE Trans Netw Serv Manag 6(3):175–189
51. Mason L (1973) An optimal learning algorithm for s-model environments. IEEE Trans Autom Control 18(5):493–496
52. Beigy H, Meybodi MR (2006) Utilizing distributed learning automata to solve stochastic shortest path problems. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 14(05): 591–615
53. Torkestani JA, Meybodi MR (2010) An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata. Comput Netw 54(5):826–843
54. Torkestani JA, Meybodi MR (2012) Finding minimum weight connected dominating set in stochastic graph based on learning automata. Inf Sci 200:57–77

55. Torkestani JA, Meybodi MR (2012) A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs. J Supercomput 59(2):1035–1054
56. Lü Z, Hao J-K (2012) Adaptive memory-based local search for max-sat. Appl Soft Comput 12(8):2063–2071
57. Bouhmala N, Groesland MS, Volden-Freberg V (2016) Enhanced metaheuristics with the multilevel paradigm for max-csps, in: International Conference on Computational Science and Its Applications, Springer, pp. 543–553
58. Bouhmala N (2012) A multilevel memetic algorithm for large sat-encoded problems. Evol Comput 20(4):641–664
59. Selman B, Levesque HJ, Mitchell DG et al. (1992) A new method for solving hard satisfiability problems., in: AAAI, Vol. 92, pp. 440–446
60. Selman B, Kautz HA, Cohen B (1994) Noise strategies for improving local search, in: AAAI, Vol. 94, pp. 337–343
61. McAllester D, Selman B, Kautz H (1997) Evidence for invariants in local search, in: AAAI/IAAI, Rhode Island, USA, pp. 321–326
62. Glover F (1989) Tabu search"part i". ORSA J Comput 1(3):190–206
63. Hansen P, Jaumard B (1990) Algorithms for the maximum satisfiability problem. Computing 44(4):279–303
64. Gent IP, Walsh T (1995) Unsatisfied variables in local search, Hybrid problems, hybrid solutions 73–85
65. Gent IP, Walsh T (1993) Towards an understanding of hill-climbing procedures for sat, in: AAAI, Vol. 93, pp. 28–33
66. Cha B, Iwama K (1995) Performance test of local search algorithms using new types of random cnf formulas, in: IJCAI, Vol. 95, pp. 304–310
67. Frank J (1997) Learning short-term weights for gsat, in: IJCAI (1), pp. 384–391
68. Spears WM (1993) Simulated annealing for hard satisfiability problems., in: Cliques, Coloring, and Satisfiability, Citeseer, pp. 533–558
69. Bouhmala N (2019) Combining simulated annealing with local search heuristic for max-sat. J Heuristics 25(1):47–69
70. Eiben A, Van der Hauw J (1997) Solving 3-sat with adaptive genetic algorithms, in: Proceedings of the 4th IEEE Conference on Evolutionary Computation, Vol. 81, IEEE Press, p. 86
71. Johnson DS, Trick MA (1996) Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11–13, 1993, Vol. 26, American Mathematical Soc
72. Hao J-K, Lardeux F, Saubion F (2003) Evolutionary computing for the satisfiability problem, in: Workshops on Applications of Evolutionary Computation, Springer, pp. 258–267
73. Smyth K, Hoos HH, Stützle T (2003) Iterated robust tabu search for max-sat, in: Conference of the Canadian Society for Computational Studies of Intelligence, Springer, pp. 129–144
74. Kar AK (2016) Bio inspired computing–a review of algorithms and scope of applications. Expert Syst Appl 59:20–32
75. Thathachar MAL, Sastry PS, A new approach to designing reinforcement schemes for learning automata, IEEE Transactions on Systems, Man, and Cybernetics SMC-15
76. Gutjahr WJ (2002) Aco algorithms with guaranteed convergence to the optimal solution. Inf Process Lett 82(3):145–153
77. Hoos HH (2002) An adaptive noise mechanism for walksat, in: Eighteenth national conference on Artificial intelligence, American Association for Artificial Intelligence, pp. 655–660
78. Taillard É (1991) Robust taboo search for the quadratic assignment problem. Parallel Comput 17(4–5):443–455
79. Li CM, Wei W, Zhang H (2007) Combining adaptive noise and look-ahead in local search for sat, in: International Conference on Theory and Applications of Satisfiability Testing, Springer, pp. 121–133
80. Tompkins DA, Hoos HH (2004) Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat, in: International conference on theory and applications of satisfiability testing, Springer, pp. 306–320
81. Wauters T, Verbeeck K, De Causmaecker P, Berghe GV (2013) Boosting metaheuristic search using reinforcement learning, in: Hybrid Metaheuristics, Springer, pp. 433–452
82. Martello S (ed) (1985) Survey in combinatorial optimization. Elsevier North-Holland, Inc., New York
83. Santharam G, Sastry PS, Thathachar MAL (1994) Continuous action set learning automata for stochastic optimization. Journal of the Franklin Institute 331B5:607–628

Anis Yazidi received the M.Sc. and Ph.D. degrees from the University of Agder, Grimstad, Norway, in 2008 and 2012, respectively. He was a Researcher with Teknova AS, Grimstad, Norway. He is currently a Full Professor with the Department of Computer Science, Oslo Metropolitan University, Oslo, Norway, where he is leading the research group in Applied Artificial Intelligence. His current research interests include machine learning, learning automata, stochastic optimization, and autonomous computing.

Noureddine Bouhmala received his PhD from the University of Neuchatel, department of computer science, Switzerland. In 2001, he joined the department of computer science at the University of SouthEast Norway. Since 2007 he has been affiliated with the University of Agde in 20% position at the department of information and communication. His research includes combinatorial optimization, data mining, algorithms, artificial intelligence. Much of his work has been on improving the understanding, design, performance of large scale optimization problems.

Morten Goodwin is an Associate Professor at the University of Agder. His field of expertise is artificial intelligence, particularly swarm intelligence and neural networks. Morten Goodwin received the B.Sc. and M.Sc. degrees from the University of Agder, Norway, in 2003 and 2005, respectively, and the Ph.D. degree from Aalborg University Department of Computer Science, Denmark, in 2011, with on applying machine learning algorithms on eGovernment indicators which are difficult to measure automatically. He is an Associate Professor with the Department of ICT, the University of Agder, deputy director for Centre for Artificial Intelligence Research, coordinator for the International Master's Programme in ICT, a public speaker, and an active researcher. His main research interests include machine learning, swarm intelligence, deep learning, and adaptive learning in the fields of accounting, medicine, games, and chatbots. He has more than 90 peer-reviewed scientific publications in the area.