Reinforcement learning in learning automata and cellular learning automata via multiple reinforcement signals

Reza Vafashoar, Mohammad Reza Meybodi
Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran.
Tel: +98-21-64545120; Fax: +98-21-66495521; e-mail: (vafashoar, mmeybodi)@aut.ac.ir

Abstract

Many scientific and engineering problems are decentralized in nature. Various distributed approaches have been developed for solving these problems, and among them, cellular learning automaton has demonstrated to be an effective model for systems consisting of a large number of interacting components. In the cellular learning automata approach, each such component is modeled by a learning automaton. The learning automaton associated with a component aims to learn the action which best suites with its neighboring components. This objective becomes more challenging when the automaton is required to find the optimal subset of its available actions. The common learning automata algorithms can deal with this problem by considering all combinations of their allowable actions as new action sets. However, this approach is only applicable for small action spaces. The current work extends some common learning automata algorithms so that they can efficiently learn the optimal subset of their actions through parallel reinforcements. These parallel reinforcements represent the favorability of each action in the performed subset of actions; consequently, the learning automaton would be able to learn the effectiveness of each action individually. By integrating the new LA models in a cellular learning automaton, each component of the system is able to interact with its neighbors simultaneously via multiple actions. In order to investigate the effectiveness of the proposed models, their applicability on a channel assignment problem is investigated experimentally. The achieved results demonstrate the efficiency of the proposed multi-reinforcement learning schemes.

Keywords: Learning automata, Cellular learning automata, Multi radio channel assignment

1. Introduction

Learning automata belong to a category of machine learning algorithms which is known as reinforcement learning (RL) [1-2]. RL is about agents operating in unknown random environments trying to make adaptive decisions through a trial and error procedure. A learning automaton (LA) is equipped with a set of actions, and interacts with an environment. During each interaction, the LA chooses an action according to its internal probability distribution. Then, the environment triggers a response which represents the acceptance level of the selected action. Following the response of the environment, the LA updates its internal probability distribution to adapt to the environment. The goal of the LA is to choose the optimal action which results in the most favorable environmental responses.

LA provides a means for developing appropriate decision policies with minimal domain information. It generally does not require environmental dynamics to be known beforehand. Due to these characteristics, LAs have found a broad range of applications [3-6]. Varieties of LA models have been investigated in the literature, which have been shown to converge to the desired solutions under appropriate conditions. Zhang et al. developed a fast discretized pursuit LA and provided proofs on its ε-optimality [7]. Their model has demonstrated an interesting convergence speed. Another motivating ε-optimal LA is the last-position elimination-based learning automaton (LELA), which gradually eliminates the less reward receiving actions [8]. Approaches like stochastic discretized weak estimator (SDWE), which is based on the principles of discretized LA, are able to track time-varying distributions [9].

Another closely related RL method to LA is Q-learning [10].The goal of Q-Learning is to learn an optimal policy for an agent, which guides its action selection under different circumstances. Like LA, a Q-learning agent operates in an unknown random environment and learns its optimal policy through interactions with the environment. The execution of each action may provoke the environment to change its current state. Accordingly, the goal of the Q-learning algorithm is to maximize its total future reward. To attain this, in each time step, Q-learning also considers the expected payoff from future states as well as its current state.

Hybrid approaches based on RL have been developed to tackle large state spaces [11], large action spaces [12-13], and distributed and decentralized problems [14]. The field has enjoyed a great deal of progress in the recent years and become one of the hot topics within disciplines like deep learning. Combining RL and deep learning techniques enables an RL agent to have a good perception of its environment and makes it possible to learn abstractions over high-dimensional state spaces [11-13]. In contrast to many classical deep learning approaches that usually focus on perception, deep RL adds the dimension of actions that actually affect the environment. Deep RL has been successfully applied to various domains such as games [11], visual object tracking [15], autonomous control [16-18], handwritten digit recognition [19], biological data mining [20], and cloud computing and residential smart grid systems [21].

Interconnected LA (RL in general) agents have given rise to new computational models suitable for decentralized problems [14]. Cellular learning automaton (CLA), introduced by Beigy and Meybodi, is a hybrid mathematical model for many decentralized problems [22-26]. It can be considered as a network of agents with learning abilities. The interaction among the agents is guided by the CLA rules, and these agents use the learning properties of the LA to adapt their behavior. Up to now, various CLA models including open CLA [22], asynchronous CLA [14], irregular CLA [25], associative CLA [27], dynamic irregular CLA [28], and wavefront CLA [29] have been successfully employed in the areas such as peer-to-peer networks [30-31], numerical optimization [32-33], detecting community structures in complex networks [34], stochastic graphs [35], social networks [36-37], distributed task allocation [38], and wireless sensor networks [39].

As stated previously, the operation of a typical LA can be viewed as a sequence of iterative feedback cycles in which the LA interacts with its environment. During each cycle, the LA selects an action, which triggers a reinforcement feedback from the environment. This action can be a vector (or a set) of simpler components [40]. Herein, we use the terms super-action and simple-action to distinguish between a vector action and its comprising components. The reinforcement signal received in response to a super-action represents the favorability of a combination of simple-actions. When an action is a subset of simple-actions, the number of choices to be learned by the LA becomes exponential in the number of available simple-actions. Very few works have been reported on the optimal subset selection problem in the LA literature. Zhang et al. developed several methods based on pursuit LA for the selection of optimal subsets [41]. Considering an action set consisting of N simple-actions, their algorithm uses a specific updating procedure in order to learn an optimal subset of size w. During each step, the algorithm decreases the selection probability of N-w simple-actions with the lowest estimated rewards. Then, it calculates the probability sum of these N-w simple-actions and equally distributes the remaining probability amount to the other w simple-actions with higher reward estimates. This approach may be problematic as the probabilities of simple-actions do not change in an orderly fashion: the new probability values of the estimated best simple-actions are obtained independent of their previous values. Additionally, no evidence on the convergence behavior of the introduced algorithms is provided. Later, Guo et al. introduced other models of LA which are capable of subset selection [42]. However, their introduced models suffer from very similar issues.

Sometimes the reinforcement feedback from the environment can be defined as a combination of distinct signals, where each signal represents the effectiveness of each simple-action in a selected super-action. In this case, an RL agent can effectively learn the value of each simple-action individually. In this paper, we extend the LA models so that they can learn the optimal subsets in an appropriate time given that the environment provides each selected simple-action in a super-action with an individual reinforcement signal. The introduced LA models can also be integrated into the CLA approach. In order to study the characteristics of multi-reinforcement CLA, a new general CLA model is also developed in this paper. In this new model, each LA selects a set of actions at each step and then receives a reinforcement vector from its environment. Based on the received reinforcement vector, each cell adjusts its internal parameters. The new CLA model is an extension of the model previously introduced in [43]. However, it can learn to select a subset of actions rather than a single action. We analyze the convergence properties of the proposed multi-reinforcement CLA for a class of local rules called potential local rules, which is related to potential games in the game theory [44].

In order to illustrate the applicability of the proposed models, they will be utilized on a channel assignment problem. In the considered problem, each router can be equipped with multiple radio interfaces that operate on multiple non-overlapping channels. Consequently, the network nodes can utilize the radio spectrum more efficiently. Two close communications can happen if they are conducted on different non-overlapping channels. With limited number of channels and radio interfaces, the objective of a channel assignment is to minimize the total network interference.

2 Preliminaries

In this section, we briefly review the fundamental concepts in learning automata, cellular automata, and cellular learning automata.

2.1. Learning automaton

A variable structure LA can be represented by a sextuple like $\{\Phi, \alpha, \beta, A, G, P\}$ [1], where Φ is the set of internal states; α is the set of outputs or actions of the LA; β is the set of inputs or environmental responses; A is the learning algorithm; $G(.):\Phi\rightarrow\alpha$ is the function that maps the current state into the current output; and P is the probability vector that determines the selection probability of a state at each step. Usually, there is a one to one correspondence between Φ and α; as a results, the two terms can be used interchangeably.

The objective of an LA is to identify the best action which maximizes the expected received payoff from the environment [2]. To attain this goal, the LA selects an action according to its action probability distribution and applies this action to the environment. The environment measures the favorability of the received action, and responds the LA with a noisy reinforcement signal β. The LA uses this response to adjust its internal action probabilities via its learning algorithm. We assume that the environmental response to a selected action like a_i at step k is $\beta\in\{0,1\}$, where 0 and 1 are used to represent pleasant and unpleasant responses, respectively. The internal action probabilities can be updated according to Eq. (1) when the response is pleasant and according to Eq. (2) when it is not.

$$p_j(k+1) = \begin{cases} p_j(k) + a(1 - p_j(k)) & \text{if } i = j \\ p_j(k)(1-a) & \text{if } i \neq j \end{cases}. \qquad (1)$$

$$p_j(k+1) = \begin{cases} p_j(k)(1-b) & \text{if } i = j \\ \dfrac{b}{r-1} + p_j(k)(1-b) & \text{if } i \neq j \end{cases}. \qquad (2)$$

In these two equations, *a* and *b* are called reward and penalty parameters, respectively. For *a*=*b* the learning algorithm is called $L_{R\text{-}P}$; when *b* << *a*, it is called $L_{R\varepsilon P}$; and when *b* is zero, it is called $L_{R\text{-}I}$.

Before we continue, the following norms of behavior are defined formally.

ε-optimality: an LA is said to be ε-optimal if

$$\lim_{k\to\infty} \inf p_l(k) > 1 - \varepsilon \ w.p.1 \quad (3)$$

can be achieved for any ε>0 by a proper choice of the parameters in the LA. Here, $p_l(k)$ is the selection probability of the optimal action at step *k* [2].

Absolute expediency: an LA is said to be absolutely expedient if

$$E[M(k+1)|p(k)]>M(k) \qquad (4)$$

for all *k*. Here, *M*(*k*) is the average reward for the given probability vector at step *k* [1].

2.2. Pursuit algorithm

Pursuit belongs to a class of RL methods called estimator algorithms. It uses the history of the previously selected actions and the obtained reinforcements to estimate the reward probability of different actions. The objective of the pursuit algorithm, similar to other RL approaches, is to select the optimal action with the probability one. However, as the estimated rewards could be erroneous, the algorithm gradually increases the selection probability of the most promising action. In order to increase the probability of the best-estimated action, the pursuit algorithm first decreases the probabilities of all actions. Then, it adds the deducted probability mass to the probability of the best-estimated action.

The reward estimate of the *i*th action of an LA at step *k* is denoted by $d_i(k)$. In order to calculate the reward estimates, pursuit maintains two vectors $[\eta_1(k),\ldots, \eta_r(k)]^T$ and $[Z_1(k),\ldots, Z_r(k)]^T$. Here, $\eta_i(k)$ gives the number of times that the *i*th action is chosen till step *k* and $Z_i(k)$ gives the number of times that this action is rewarded till step *k*. The operation of pursuit involves updating these vectors. During step *k*, the LA selects an action like $\alpha(k)=a_i$ according to its internal probability vector *p*(*k*). Then, the environment generates a reinforcement signal $\beta(k)$ in response to this selected action. Next, the LA uses $\alpha(k)$ and $\beta(k)$ to update its variables and obtains new reward estimates according to Eq. (5). Finally, the action probabilities are updated according to Eq. (6).

$$\begin{aligned} Z_i(k+1) &= Z_i(k) + (1-\beta(k)) \\ \eta_i(k+1) &= \eta_i(k) + 1 \\ d_i(k+1) &= \frac{Z_i(k+1)}{\eta_i(k+1)} \end{aligned} \quad , \qquad (5)$$

$$p_j(k+1) = \begin{cases} p_j(k) - \lambda p_j(k) + \lambda & \text{if } d_j(k+1) = \max_l \left(d_l(k+1)\right) \\ p_j(k) - \lambda p_j(k) & \text{if } d_j(k+1) < \max_l \left(d_l(k+1)\right) \end{cases}, \quad (6)$$

where λ is the learning rate of the LA.

At the beginning, each action is chosen for a few times to obtain initial reward estimates. Also, we can simply initialize *Z*(0) and $\eta(0)$ using some constant values.

2.3. Cellular automaton

Cellular automata (CAs) [45] are abstract models for systems consisting of a large number of identical simple components. These components are called cells in the CA terminology and are usually arranged in some regular forms such as grid. A CA operates in discrete times called steps. At each time step, the cells instantiate one of a finite set of states, and the states of all cells together define the state of the cellular automaton. During a time step, a local rule defines the state of a cell for the next step. The local rule defines the state of a cell for the next time step based on the current states of its neighboring cells. Many neighborhood patterns are suggested for a CA such as Moore and von Neumann. In spite of its simplicity, CA can produce complicated patterns of behavior; as a result, CA has become a popular tool for modeling complicated systems and natural phenomena.

2.4. Cellular learning automaton

A cellular learning automaton is a combination of a cellular automaton with learning automata [26]. In this model, each cell of a CA contains one or more LAs. The LA or LAs residing in a particular cell define the state of the cell. Like CA, a local rule controls the behavior of each cell. At each time step, the local rule defines the reinforcement signal for a particular LA based on its selected action and the actions chosen by its neighboring LAs. Consequently, the neighborhood of each LA is considered to be its local environment. A formal definition of CLA is provided by Beigy and Meybodi [23, 26]. The authors have also investigated the asymptotic behavior of the model and provided some theoretical analysis on its convergence.

The structure of a CLA can be represented by a graph where each vertex denotes a CLA cell. Each edge of this graph defines a neighborhood relation between its two incident nodes. We represent the action set of the i^{th} LA by $A_i = \{a_{i1},...,a_{im_i}\}$, where m_i is the number of the LA actions. The probability vector of the i^{th} LA, which determines the selection probability of its actions, is denoted by p_i. This probability vector defines the internal state the LA, and the probability vectors of all learning automata in the CLA define the configuration of the CLA at each step k. The configuration of a CLA at step k is denoted by $P(k) = (p_1(\mathrm{k}), p_2(\mathrm{k}), \ldots, p_n(k))^T$, where n is the number of cells. The operation of a CLA is governed by a set of local rules. At each step k, the local rule of the i^{th} cell determines the reinforcement signal to the learning automaton in the cell as follows:

$$\begin{aligned} &F_i : \varphi_i \rightarrow \underline{\beta} \\ &with \\ &\varphi_i = \prod\nolimits_{j=0}^{n_i} A_{N_i(j)} \end{aligned} \quad , \quad (7)$$

where N_i is the neighborhood function of the i^{th} cell, and $N_i(j)$ returns the index of the j^{th} neighboring cell to cell i with $N_i(0)$ defined as $N_i(0)=i$. n_i denotes the number of cells in the neighborhood of the i^{th} cell. Finally, $\underline{\beta}$ is the set of values that a reinforcement signal can take. We can extend the local rules of a CLA as follows:

$$T_i\left(\alpha_1,...,\alpha_n\right) = F_i\left(\alpha_i, \alpha_{N_i(1)},...,\alpha_{N_i(n_i)}\right) \ \forall \alpha_j \in A_j \ . \quad (8)$$

The expected reward of each LA at each step can be obtained based on its associated local rule and the CLA configuration. The expected reward of an action like $a_r \in A_i$ in configuration P is denoted by $d_{ir}(P)$. A configuration P is called compatible if the following condition holds for any configuration Q and any cell i in the CLA.

$$\sum\nolimits_r d_{ir}(P) \times p_{ir} \geq \sum\nolimits_r d_{ir}(Q) \times q_{ir} \ . \quad (9)$$

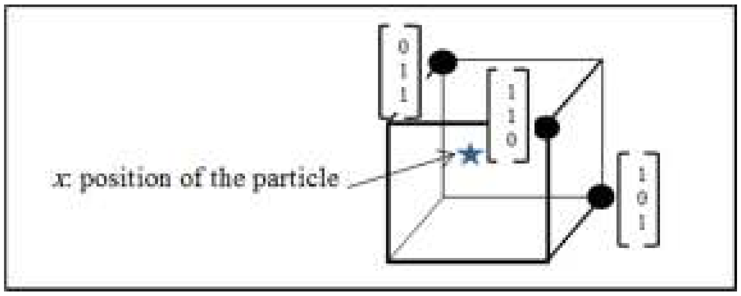
The definition of compatible point is equivalent to Nash equilibrium in game theory [43]. In game theory, a game is said to be a potential game if the change in the utility of a player for a switch in her strategy can be expressed using a single global function. A local rule will be called a potential local rule if there exists a function like ψ, called a potential function, for which the condition in Eq. (10) is satisfied. The characteristics of local utility functions in potential games have been investigated in several recent works, and interested readers can refer to [46-47].

$$\begin{aligned} &T_i\left(\alpha_1,\ldots,\alpha_i,\ldots,\alpha_n\right)-T_i\left(\alpha_1,\ldots,\alpha_i{}',\ldots,\alpha_n\right)= \\ &\psi\left(\alpha_1,\ldots,\alpha_i,\ldots,\alpha_n\right)-\psi\left(\alpha_1,\ldots,\alpha_i{}',\ldots,\alpha_n\right)\ \forall\alpha_j\in A_j,\ \forall\alpha_i{}'\in A_i \end{aligned}. \quad (10)$$

A CLA starts from some initial state (it is the internal state of every cell). At each stage, each automaton selects an action according to its probability vector and performs it in its local environment. Next, the rule of the CLA determines the reinforcement signals (the responses) to the LAs residing in its cells, and based on the received signals, each LA updates its internal probability vector. This procedure continues until a termination condition is satisfied.

3. Reinforcement learning using multiple reinforcements

This section introduces LA models that can learn an optimal subset of their actions and update their internal probability vectors according to multiple environmental responses. In order to distinguish between a vector action and its comprising components, the terms super-action and simple-action are adopted in this work. It is assumed that each of the proposed LAs has r simple-actions and aims at learning the optimal super-action which consists of w simple-actions ($1\leq w<r$). In each time step, the LA selects a super-action consisting of w simple-actions. Then, it performs the combination of these simple-actions as a single super-action in the environment. The environment provides a w-dimensional reinforcement vector in response to the selected super-action. Each element of the reinforcement vector represents the favorability of one of the selected simple-actions in the combined super-action. Accordingly, each selected simple-action like $\alpha_j(k)$ receives an independent reinforcement signal like $\beta_j(k)$. The combined reinforcement for the selected super-action at step k can be considered as the sum of all components of the reinforcement vector, i.e. $\sum_{l=1}^{w}\beta_l(k)$. The goal of the learning algorithm is to learn the super-action with the highest expected combined reinforcement value.


$\begin{bmatrix}0\\1\\1\end{bmatrix}$
$\begin{bmatrix}1\\1\\0\end{bmatrix}$
$\begin{bmatrix}1\\0\\1\end{bmatrix}$
x: position of the particle
Fig. 1. The internal state of a multi-reinforcement LA which aims at finding the optimal subset of size two out of three actions. The energy level of the LA is modeled as particle which flies in a 3-dimensional hypercube.

3.1. Multi-reinforcement learning automaton I

The internal state of an LA is ordinarily represented as a probability vector. During the learning process, this probability vector is adapted to the environment, and the selection probability of the most reward receiving action is gradually increased. However, using this method for subset selection would be inefficient as the number of super-actions grows exponentially with the number of simple-actions. Additionally, the reinforcements received by a typical simple-action depend on the effectiveness of other simple-actions in the experienced super-actions, which adversely affects the learning process. Representing the internal state as a probability vector over simple-actions would also be unsuitable: if the selection probability of a simple-action converges to one, the selection probabilities of all the other simple-actions converge to zero.

In order to deal with the aforementioned issues, we propose two approaches. In the first approach, an r-dimensional energy vector is considered for an LA, and the internal state of the LA is modeled as a particle, which moves in an r-dimensional hypercube (Fig. (1)). The position of the particle in the hyper-cube represents the energy level of each simple-action. Actions with higher energy levels are selected with higher probabilities. The automaton in Fig. (1) has three actions and selects a super-action consisting of two simple-actions in each step. These super-actions correspond to the hypercube corners which are represented by the black circles in the figure. The objective of the proposed LA is to move the particle toward the hypercube corner of the energy with the highest expected payoff.

We assume that the energy level of the LA particle (its position in the LA hypercube) at step k is represented by $x(k)$, where $x_i(k)$ corresponds to the energy level of the i^{th} simple-action. At each stage of the first developed algorithm, a dummy super-action is selected based on the energy vector $x(k)$: each simple-action a_i is chosen with probability $x_i(k)$ in the dummy super-action. This super-action corresponds to one of the corners of the LA hypercube and may consist of less/more than w simple-actions. After this stage, the algorithm randomly selects one of the closest w-dimensional super-actions to the selected dummy super-action. Accordingly, the final selected super-action (which is denoted by $\alpha(k)$) consists of exactly w simple-actions. It should be noted that if we consider each element of $x(k)$ as the energy level of a simple-action, the final super-action ($\alpha(k)$) is selected based on the energy levels of the simple-actions. After action selection, the selected super-action is carried out in the environment. Then, the LA receives a w-dimensional reinforcement signal, which represents the favorability of each one of the selected simple-actions. Based on this signal, the estimated rewards of the simple-actions are updated according to Eq. (11). The algorithm maintains two additional vectors $Z_i(k)$ and $\eta_i(k)$ in order to compute the estimated rewards of the simple-actions. $Z_i(k)$ represents the total received reinforcement by the ith action, and $\eta_i(k)$ represents the number of times that the ith action is selected till step k.

$$\begin{aligned}
Z_i(k) &= \begin{cases} Z_i(k-1)+\beta_i(k) & \text{if } a_i \in \alpha(k) \\ Z_i(k-1) & \text{otherwise} \end{cases} \\
\eta_i(k) &= \begin{cases} \eta_i(k-1)+1 & \text{if } a_i \in \alpha(k) \\ \eta_i(k-1) & \text{otherwise} \end{cases}, \qquad (11) \\
\widehat{d}_i(k) &= \frac{Z_i(k)}{\eta_i(k)},\ i=1,\dots,r
\end{aligned}$$

where $\widehat{d}_i(k)$ represents the estimated reward for the ith simple-action at step k. Additionally, for simplicity, the reinforcement vector is represented by an r-dimensional vector; however, only w elements of this vector corresponding to the selected simple-actions have valid information.

The next step of the algorithm is to find w simple-actions with the highest estimated rewards. This set of simple-actions corresponds to the super-action with the highest expected reward. Then, the algorithm moves the LA particle toward the corner of the LA hypercube associated with this super-action. Consequently, the energy levels of the promising simple-actions are increased. Fig. (2) represents the pseudocode of the proposed multi reinforcement LA, and a block diagram for the proposed method is provided in Fig. (3). It should be noted that the algorithm can be simply implemented without actually calculating C (in Fig. (2)). Consequently, all of its steps can be efficiently implemented with computational complexities not exceeding o(r).

```
Algorithm I
Input
$r$: number of actions
$w<r$: number of actions to be selected.
$\lambda$: learning rate.
Initialization:
$Z_i(0) = c_1\ \forall i\in\{1,\ldots,r\}$, where $c_1$ is a constant value.
$\eta_i(0) = c_2\ \forall i\in\{1,\ldots,r\}$, where $c_2$ is a constant value.
$\hat{d}_i(0) = 0\ \forall i\in\{1,\ldots,r\}$.
Set $x(0)=[0.5,\ldots,0.5]^T$, where $x$ has $r$ dimensions, and $x_i\in[0,1]$ represents the energy of the $i^{th}$ action.
Let $I$ be an $r$-dimensional Boolean vector, $I_i\in\{0,1\}$.
Let $C=\{C_1,C_2,\ldots,C_o\}$ be the set of corners (vertices) of the LA hypercube with exactly $w$ 1s and $r$-$w$ 0s, where
$o=|C|=\binom{r}{w}$, $C_{ij}\in\{0,1\}$, and $\sum_j C_{ij}=w\ \ \forall i\in\{1,\ldots,o\}$.
Set $k = 0$.
repeat
1.  Set $k = k+1$ and $\alpha(k)=\{\}$.
2.  Select a random set of simple-actions according to the energy level of the LA (i.e. according to $x$): each
    action like $a_i$ is selected with probability $x_i$.
3.  For each $i\in\{1,\ldots,r\}$, set $I_i=1$ if action $a_i$ is chosen in step 2, otherwise set $I_i=0$.
4.  Let $C'$ be the subset of vertices in $C$, which are in the minimum distance to $I$. Randomly set $I$ to one of the
    vertices in $C'$.
5.  For each $i\in\{1,\ldots,r\}$, add simple-action $a_i$ to $\alpha(k)$ if $I_i=1$.
6.  Apply $\alpha(k)$ to the environment.
7.  Receive an $r$-dimensional reinforcement signal $\beta$, where $\beta_j$ is undefined if $a_j\notin\alpha(k)$.
8.  Update LA information as follows:
        $Z_i(k)=\begin{cases} Z_i(k-1)+\beta_i(k) & \text{if } a_i\in\alpha(k) \\ Z_i(k-1) & \text{otherwise}\end{cases}$
        $\eta_i(k)=\begin{cases} \eta_i(k-1)+1 & \text{if } a_i\in\alpha(k) \\ \eta_i(k-1) & \text{otherwise}\end{cases}$
        $\hat{d}_i(k)=\dfrac{Z_i(k)}{\eta_i(k)},\ i=1,\ldots,r$
9.  Update $x$ as follows:
10. Find $w$ actions with the largest estimated average reinforcements, represent them by $I$:
        $\left(I_j\in\{0,1\}\ \forall j\in\{1,\ldots,r\}\right)$ and $\sum_j I_j=w$
        and $\forall i,j\ \left(\left(I_i=1 \text{ and } I_j=0\right)\Rightarrow \hat{d}_i\geq\hat{d}_j\right)$
11. Let $C_i$ be the vertex in $C$ corresponding to $I$.
12. $x(k) = x(k-1) + \lambda(C_i-x)$.
until some stopping condition is satisfied
```
Figure 2. Pseudocode of multi-reinforcement learning automata I.

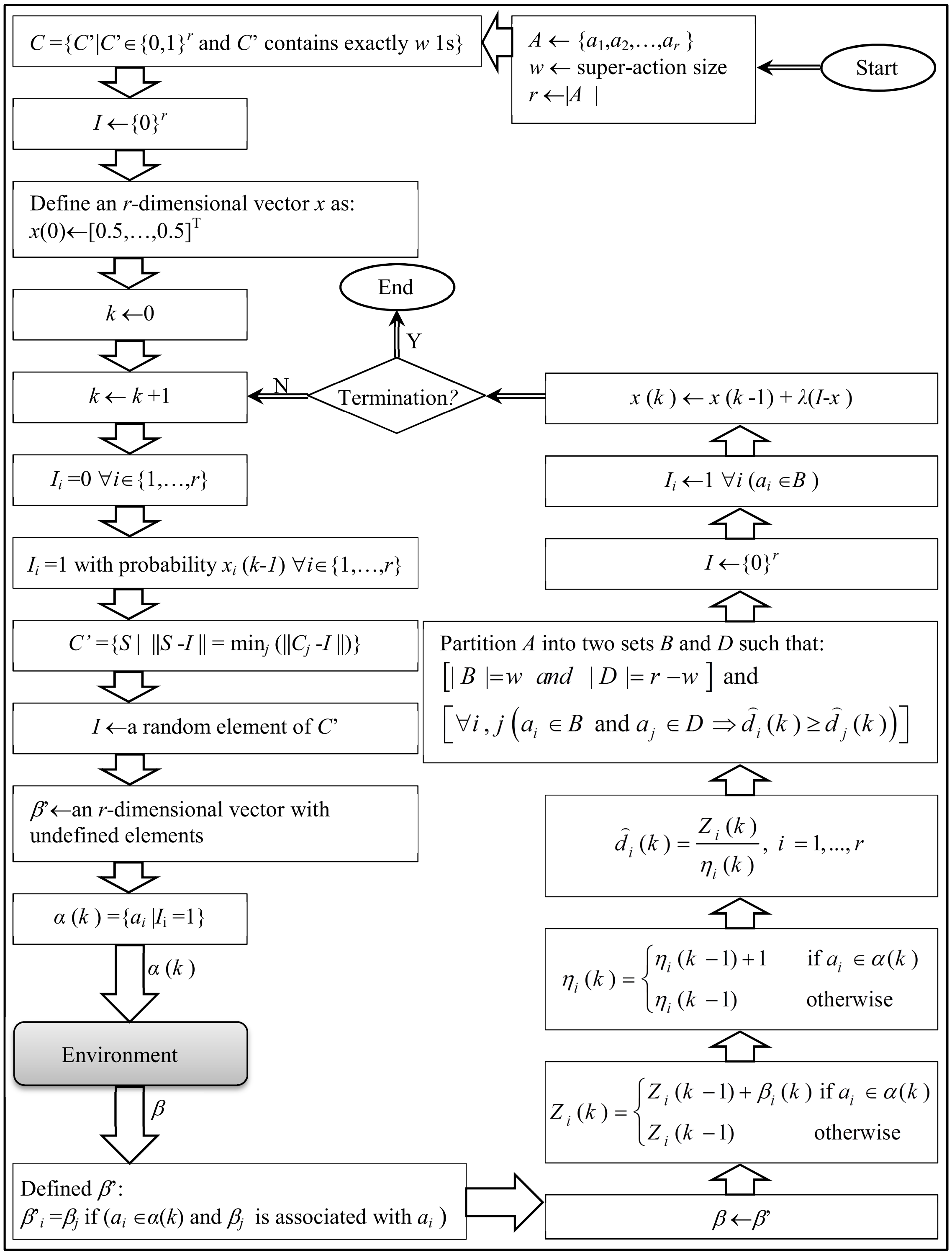

Start
$A \leftarrow \{a_1,a_2,\ldots,a_r\}$
$w \leftarrow$ super-action size
$r \leftarrow |A|$
$C = \{C' \mid C' \in \{0,1\}^r$ and C' contains exactly w 1s$\}$
$I \leftarrow \{0\}^r$
Define an r-dimensional vector x as:
$x(0) \leftarrow [0.5,\ldots,0.5]^T$
$k \leftarrow 0$
$k \leftarrow k+1$
$I_i = 0 \ \forall i \in \{1,\ldots,r\}$
$I_i = 1$ with probability $x_i(k-1) \ \forall i \in \{1,\ldots,r\}$
$C' = \{S \mid \|S - I\| = \min_j(\|C_j - I\|)\}$
$I \leftarrow$ a random element of C'
$\beta' \leftarrow$ an r-dimensional vector with undefined elements
$\alpha(k) = \{a_i \mid I_i = 1\}$
$\alpha(k)$
Environment
β
Defined β':
$\beta'_i = \beta_j$ if ($a_i \in \alpha(k)$ and β_j is associated with a_i)
$\beta \leftarrow \beta'$
$Z_i(k) = \begin{cases} Z_i(k-1) + \beta_i(k) & \text{if } a_i \in \alpha(k) \\ Z_i(k-1) & \text{otherwise} \end{cases}$
$\eta_i(k) = \begin{cases} \eta_i(k-1) + 1 & \text{if } a_i \in \alpha(k) \\ \eta_i(k-1) & \text{otherwise} \end{cases}$
$\hat{d}_i(k) = \frac{Z_i(k)}{\eta_i(k)}, \ i = 1,\ldots,r$
Partition A into two sets B and D such that:
$[|B| = w \ \ and \ \ |D| = r - w]$ and
$[\forall i, j \, (a_i \in B$ and $a_j \in D \Rightarrow \hat{d}_i(k) \geq \hat{d}_j(k))]$
$I \leftarrow \{0\}^r$
$I_i \leftarrow 1 \ \forall i \, (a_i \in B)$
$x(k) \leftarrow x(k-1) + \lambda(I - x)$
Termination?
N
Y
End

Figure 3. Block diagram of multi-reinforcement learning automata I

3.1.1. Convergence analysis of the multi-reinforcement learning automata I.

In what follows, we discuss the convergence properties of the proposed multi-reinforcement LA. In this regard, the approach given in [2] will be employed, and it will be shown that the LA particle (LA energy level) converges to the corner with the highest expected reward in the LA hypercube. In order to proceed, we assume that $\lambda(k)=1-\gamma^{1/k}$ with $\gamma\in(e^{-1}, 1)$.

Lemma 1. Given any $\delta\in(0,1]$ and $N\in\mathbb{Z}^{+}$, for each simple-action a_i, there exists $K_1(N,\delta)$ such that $\Pr\{\eta_i(k)<N\}<\delta\,\forall k> K_1(N,\delta)$. (12)

Eq. (12) is equivalent to $\sum_{s=1}^{N}\Pr\{\eta_i(k)=s\}\le\delta$, which is satisfied if

$$\Pr\{\eta_i(k)=s\}<\frac{\delta}{N}\forall s\in\{1,...,N\}. \quad (13)$$

Considering line 12 of the proposed algorithm in Fig. (2), the energy of each simple-action can decrease with a rate not larger than (1-$\lambda(k)$) at each step k. Considering line 2 of the algorithm, each action is selected according to its energy level in the first phase of the action selection and is added to I . The selection probability of any action like a_i in this phase is x_i. However, I may contain more than w actions, and accordingly some of them may be removed randomly according to line 4 of the algorithm. The removal probability of any selected action at this step is less than $(r-1)/r$. Hence,

$$\Pr\{a_i\notin\alpha(k)\}\le\left(1-\frac{x_i(0)\prod_{t=1}^{k}(1-\lambda(t))}{r}\right). \quad (14)$$

Accordingly,

$$\Pr\{\eta_i(k)=s\}<k^{s}\left(1-\frac{x_i(0)\prod_{t=1}^{k}(1-\lambda(t))}{r}\right)^{k-s}. \quad (15)$$

Now the lemma can be proved if there exists K_1 such that

$$k^{s}\left(1-\frac{x_i(0)\prod_{t=1}^{k}(1-\lambda(t))}{r}\right)^{k-s}<\frac{\delta}{N},1\le s\le N \text{ for all } k>K_1(N,\delta). \quad (16)$$

Using $\lambda(k)$=1-$\gamma^{1/k}$, it can be observed that

$$\lim_{k\to\infty} f(k) = k^{s}\left(1-\frac{x_i(0)\prod_{t=1}^{k}\left(1-\lambda(t)\right)}{r}\right)^{k-s} = 0. \qquad (17)$$

To verify Eq. (17), we can use the harmonic sum formula $\sum_{t=1}^{k}\frac{1}{t} = \ln\left(k\right)+\xi+o\left(1/k\right)$.

By using c=$x_i(0)/r$ and the harmonic sum, Eq. (17) can be restated as

$$\lim_{k\to\infty} f(k) = k^{s}\left(1-ck^{\ln(\gamma)}\gamma^{\xi}\gamma^{o(1/k)}\right)^{k-s}. \qquad (18)$$

Consequently,

$$\lim_{k\to\infty} f(k) = \lim_{k\to\infty} e^{s\ln(k)+(k-s)\ln\left(1-ck^{\ln(\gamma)}\gamma^{\xi}\gamma^{o(1/k)}\right)} = 0. \qquad (19)$$

According to Eq. (17), for sufficiently large values of k, $f(k)$ becomes small enough and is guaranteed to be less than δ/N.

Lemma 2. Given any $\varepsilon,\delta\in(0,1]$, for all $i\in\{1,\ldots,r\}$, there exists K_2 satisfying

$$\Pr\left\{|\hat{d}_i(k)-d_i|>\varepsilon\right\}<\delta \;\forall k>K_2(\varepsilon,\delta), \qquad (20)$$

where d_i=E[$\beta_i(k)$].

Considering the algorithm in Fig. (2), the estimated reward for the ith simple-action (a_i) can be expressed as:

$$\hat{d}_i(k) = \frac{\sum_{s=1}^{\eta_i(k)}\beta_i(m_s^i)}{\eta_i(k)}, \qquad (21)$$

where m_s^i represents the time instant at which a_i is chosen for the sth time.

For each i, the sequence $\{\beta_i(m_s^i)\}$ is iid, and each of its members is bounded above by a constant like M. Considering Hoeffding's inequality and substituting $\eta_i(k)$ with N:

$$\Pr\left\{\left|\frac{\sum_{s=1}^{N}\beta_i(m_s^i)}{N}-d_i\right|>\varepsilon\right\}<2\exp\left(-\frac{2N\varepsilon^2}{M^2}\right). \qquad (22)$$

Now we define two events A and B as:

$$\begin{aligned} A &= \left\{\left|\hat{d}_i(k)-d_i\right|>\varepsilon\right\} \\ B &= \left\{\eta_i(k)>N\right\} \end{aligned}. \qquad (23)$$

The probability of event A can be obtained as follows:

$$\Pr(A)=\Pr(A\mid B)\Pr(B)+\Pr(A\mid\bar{B})\Pr(\bar{B})<\Pr(A\mid B)+\Pr(\bar{B}). \qquad (24)$$

By considering $N=\left\lceil\frac{M^2}{2\varepsilon^2}\ln\frac{4}{\delta}\right\rceil$, it follows that $\Pr(A|B)<\delta/2$, and by lemma 1

$$\Pr(\bar{B})<\frac{\delta}{2}\ \ \forall k>K_2(\varepsilon,\delta) \text{ and } \lambda(k)=1-\gamma^{1/k}\ , \qquad (25)$$

where $K_2(\varepsilon,\delta)=K_1(N,\delta/2)$.

Accordingly, it follows that $\Pr(A)<\delta/2+\delta/2=\delta$.

Theorem 1. A learning automaton using the algorithm in Fig. (2) with λ defined as $\lambda(k)=1-\gamma^{1/k}$ and $\gamma\in(e^{-1},1)$ has the following property.

Given any $\varepsilon,\delta\in(0,1)$, there exists $K^*>0$ such that

$\Pr\{|x-C_L|<\varepsilon\}>1-\delta$, (26)

where C_L corresponds to the corner of the LA hypercube with w best simple-actions.

Proof:

Considering the set of optimal simple-actions as C_L, we can define the following three events:

$$E_1(k)=\{|C_L-x(k)|<\varepsilon\}, \qquad (27)$$

$$E_2(k) = \left\{ \max_i \, |\hat{d}_i(k) - d_i| < \theta \right\}, \qquad (28)$$

$$E_3(k) = \left\{ \sup_{s \geq k} \, \max_i \, |\hat{d}_i(s) - d_i| < \theta \right\}, \qquad (29)$$

where θ=min$_l\{d_l|C_{Ll}$=1$\}$-max$_i\{d_i|C_{Li}$=0$\}$.

Using these events, for any k and K, the following inequality holds:

$$\Pr\left(E_1\left(k+K\right)\right) \geq \Pr\left(E_1\left(k+K\right) | E_3\left(K\right)\right) \Pr\left(E_3\left(K\right)\right). \qquad (30)$$

Moreover, there exists K_3 such that Pr(E_1(k+K)|E_3(K))=1 for all k>K_3:

Assuming $x(0)=[1/2,\ldots,1/2]^T$ and defining $D_i(k)=|C_{Li}-x_i(k)|$, the particle can move at most K steps in the opposite direction of C_L in each dimension. Consequently,

$$D_i(K) \leq \left(1 - D_i(0) \prod_{s=1}^{K} \left(1 - \lambda(s)\right)\right) = \left(1 - 0.5 \prod_{s=1}^{K} \left(1 - \lambda(s)\right)\right) < 1. \qquad (31)$$

After step K, since $E_3(K)$ is satisfied, the LA particle moves toward C_L in each time step. Accordingly after K_3 iterations, the following holds:

$$D_i(K + K_3) \leq D_i(K) \prod_{s=1}^{K_3} \left(1 - \lambda(s+K)\right). \qquad (32)$$

Using $\lambda(k)=1-\gamma^{1/k}$,

$$D_i(K + K_3) \leq D_i(K) \prod_{s=1}^{K_3} \gamma^{1/(s+K)}. \qquad (33)$$

In Eq. (33),

$$\prod_{s=1}^{K_3} \gamma^{1/(s+K)} = \gamma^{\sum_{s=1}^{K_3+K} \frac{1}{s} - \sum_{s=1}^{K} \frac{1}{s}}. \qquad (34)$$

By defining $\kappa = D_i(K) / \left(\gamma^{\sum_{s=1}^{K} \frac{1}{s}} \right)$, Eq. (32) can be stated as

$$D_i(K + K_3) \leq \kappa \gamma^{\sum_{s=1}^{K_3+K} \frac{1}{s}}. \qquad (35)$$

Then, it follows that

$$D_i(K+K_3) \leq \kappa\gamma^{\ln(K_3+K)}. \qquad (36)$$

$D_i(K+K_3)$ approaches zero as $K_3\to\infty$ for all $i\in\{1,\ldots,r\}$. Consequently, there exists K_3 such that $\Pr(E_1(k+K)|E_3(K))=1$ for all $k>K_3$.

Hence, the proof of Theorem 1 can be completed if there exists K_4 such that $\Pr(E_3(k))\geq 1-\delta$ for all $k>K_4$.

The probability of event $E_3(k)$ can be defined as follows:

$$\Pr\left(E_3(k)\right) = \Pr\left(\bigcap\nolimits_{s=k}^{\infty} E_2(k)\right) = 1-\Pr\left(\bigcup\nolimits_{s=k}^{\infty} \bar{E}_2(k)\right). \qquad (37)$$

By defining $m(k)$ as $m(k) = \arg\max\left(\left|\hat{d}_i(k)-d_i(k)\right|\right)$ and using $\eta'(k)=\eta_{m(k)}(k)$, it follows that

$$\Pr\left(\bigcup\nolimits_{s=k}^{\infty} \bar{E}_2(k)\right) \leq \sum\nolimits_{s=k}^{\infty} \Pr\left(\bar{E}_2(k)\right) = \sum\nolimits_{s=k}^{\infty} \sum\nolimits_{t=0}^{s}\left(\Pr\left(\bar{E}_2(s)\,|\,\eta'(s)=t\right)\Pr\left(\eta'(s)=t\right)\right). \quad (38)$$

Using Hoeffding's inequality,

$$\Pr\left(\bar{E}_2(s)\,|\,\eta'(s)=t\right) \leq 2e^{-2t\theta^2/M^2}. \qquad (39)$$

Consequently,

$$\begin{aligned}&\Pr\left(\bigcup\nolimits_{s=k}^{\infty} \bar{E}_2(k)\right) \leq \sum\nolimits_{s=k}^{\infty} \sum\nolimits_{t=0}^{s}\left(\Pr\left(\bar{E}_2(s)\,|\,\eta'(s)=t\right)\Pr\left(\eta'(s)=t\right)\right) \leq \\ &\sum\nolimits_{s=k}^{\infty} 2\sum\nolimits_{t=0}^{s}\left(e^{-2t\theta^2/M^2}\Pr\left(\eta'(s)=t\right)\right)\end{aligned}. \qquad (40)$$

The RHS of Eq. (40) can be restated as follows:

$$\sum\nolimits_{s=k}^{\infty} 2\sum\nolimits_{t=0}^{s}\left(e^{-2t\theta^2/M^2}\Pr\left(\eta'(s)=t\right)\right) = \sum\nolimits_{s=k}^{\infty} 2E\left[e^{(-2\theta^2/M^2)\eta'(s)}\right]. \quad (41)$$

Define $\eta'(k) = \sum\nolimits_{s=1}^{k}\chi(s)$, where $\chi(s)\in\{0,1\}$, and $\chi(s)=0,1$ indicates whether or not the simple-action corresponding to $m(k)$ is selected at step s. By defining $\tau=-2\theta^2/M^2$, it follows that

$$\sum_{s=k}^{\infty} 2E\left[e^{\tau\eta'(s)}\right]=\sum_{s=k}^{\infty} 2E\left[e^{\tau(\chi(1)+\chi(2)+\ldots+\chi(s))}\right]=$$
$$\sum_{s=k}^{\infty} 2\prod_{t=1}^{s}\left(1-\Pr(a_{m(s)}\in\alpha(t))+\Pr(a_{m(s)}\in\alpha(t))e^{\tau}\right)= \qquad (42)$$
$$\sum_{s=k}^{\infty} 2\prod_{t=1}^{s}\left(1-\Pr(a_{m(s)}\in\alpha(t))\left(1-e^{\tau}\right)\right).$$

Similar to lemma 1, $\Pr(a_{m(s)}\in\alpha(t))$ can be replaced by a smaller term:

$$\sum_{s=k}^{\infty} 2\prod_{t=1}^{s}\left(1-\Pr(a_{m(s)}\in\alpha(t))\left(1-e^{\tau}\right)\right)\le$$
$$\sum_{s=k}^{\infty} 2\prod_{t=1}^{s}\left(1-\left(\frac{x_{m(s)}(0)}{r}\right)\left(1-e^{\tau}\right)\gamma^{\sum_{n=1}^{s}1/n}\right)=\sum_{s=k}^{\infty} 2\left(1-\left(\frac{x_{m(s)}(0)}{r}\right)\left(1-e^{\tau}\right)\gamma^{\sum_{n=1}^{s}1/n}\right)^{s}. \qquad (43)$$

The RHS of Eq. (43) can be approximated as follows:

$$\sum_{s=k}^{\infty} 2\left(1-\left(\frac{x_{m(s)}(0)}{r}\right)\left(1-e^{\tau}\right)\gamma^{\sum_{n=1}^{s}1/n}\right)^{s}\approx 2\sum_{s=k}^{\infty}\left(1-\left(\frac{x_{m(s)}(0)}{r}\right)\left(1-e^{\tau}\right)\gamma^{\ln(s)}\right)^{s}. \qquad (44)$$

The series in the RHS of Eq. (44) is convergent. Consequently, following the convergence of partial sums, for any δ there exists a K_4 such that for $s>K_4$ the series ends up less than $\delta/2$.

We can verify that the series in Eq. (44) is convergent as follows.

Define $a=(x_{m(s)}(0)/r)(1-\exp(\tau))$ and $b=-\ln(\gamma)$. The RHS of Eq. (44) can be stated as follows:

$$f(s)=\sum_{s=1}^{\infty}\left(1-\frac{a}{s^{b}}\right)^{s}, \qquad (45)$$

where $a,b\in(0,1)$.

As $\left(1-\frac{a}{s^{b}}\right)^{s^{b}}\le e^{-a/b}$,

$$f(s)\le\sum_{s=1}^{\infty}\left(e^{-a/b}\right)^{s^{1-b}}. \qquad (46)$$

The RHS of Eq. (46) converges like a geometric series.

3.2. Multi-reinforcement learning automata II

The previously proposed multi-reinforcement LA is based on the pursuit algorithm. It has two main drawbacks, which are due to the characteristics inherent in the pursuit scheme. In order to illustrate these drawbacks, we consider the following scenario. There is an environment which can accept an action from the set $A=\{a_1,\dots,a_4\}$ as its input in each time step. The expected reward of the environment for the i^{th} action is $d_i\ \forall i\in\{1,\dots,4\}$, where $d_l>d_j\ \forall l<j$. A pursuit LA starts operating in the environment, and after k_1 updating steps, its estimated rewards and action probabilities satisfy the following relations: $\hat{d}_4(k_1)>\hat{d}_3(k_1)>\hat{d}_1(k_1)>\hat{d}_2(k_1)$ and $p_4(k_1)>p_3(k_1)>p_1(k_1)>p_2(k_1)$. As the operation of the LA continues, and a_1 collects more rewards, the following relations are satisfied: $\hat{d}_1(k_2)>\hat{d}_4(k_2)>\hat{d}_3(k_2)>\hat{d}_2(k_2)$ and $p_1(k_2)>p_4(k_2)>p_3(k_2)>p_2(k_2)$ for some k_2. Assume that after step k_2, the received reinforcements are such that the following holds: $\hat{d}_1(k_3)=\max\left(\hat{d}_i(k_3)\right)$ for any $k_3>k_2$. Accordingly, after time step k_2, the selection probability of a_1 always increases, which is favorable; however, the relative order of the selection probabilities of the other actions remains unaltered (i.e. $p_4(k_3)>p_3(k_3)>p_2(k_3)$ for any $k_3>k_2$) regardless of their average received payoffs. The second issue with the pursuit algorithm is closely related to the first one. Consider again in the previous scenario that after step k_1, the average received payoff for action a_1 starts to increase. Define k_5 as the minimum number of required steps such that the following holds: $\hat{d}_1(k_1+k_5)=\max_i\left(\hat{d}_i(k_1+k_5)\right)$. During these k_5 steps, the probability of action a_4 always increases regardless of the received rewards. After these k_5 steps, the probability of a_1 starts to increase. However, it may take a long time until $p_1>p_4$ is satisfied. These described issues may lead to slow convergence in the pursuit algorithm.

According to the first observation, the selection probabilities of actions other than the one with the highest estimated reward may be unrelated to their obtained rewards. Accordingly, it would not affect the learning process if LA selects these actions with equal probabilities. The second issue can be solved if the action with the highest estimated reward is always selected with the highest probability. The second model for the multi-reinforcement LA is designed according to these observations and is similar to the ε-greedy scheme in Q-learning [10].

Fig. (4) summarizes the steps of the proposed multi-reinforcement LA II, and the block diagram of the algorithm is provided in Fig. (5). The algorithm performs action selection in three manners: greedy, exploratory, and mutated greedy. During step k, the algorithm selects the super-action with the highest estimated reward with probability $1-\lambda(k)$. We assume that $(\lambda(k))_{k\in\mathbb{N}}$ is a decreasing sequence of numbers, which converges to zero. Hence, it is ensured that the super-action with the highest estimated reward is ultimately selected almost surely in every iteration. Additionally, each simple-action should be selected for an adequate number of times in order to ensure that the estimated rewards converge to their expected values (as discussed in lemma 2). In this regard, with probability $\lambda(k)/2$, the proposed method selects a super-action in a random manner in line 2, which guarantees that each simple-action is chosen for an adequate number of times. Finally, with probability $\lambda(k)/2$, the algorithm can utilize the mutated greedy selection strategy in each iteration. Mutated greedy selection has the benefits of both exploratory and greedy selection strategies. It greedily selects a slight perturbed version of the best estimated super-action. Consequently, by using mutated greedy selection, the algorithm can frequently select and exploit estimated better simple-actions, while experiencing all simple-actions for an adequate number of times. After action selection, the selected super-action is carried out in the environment and the estimated rewards are updated based on the received reinforcement signals.

It is easy to show that by using a decreasing sequence $(\lambda(k))_{k\in\mathbb{N}}$ which satisfies $\lambda(k)=\kappa\mu(t)$ with $\sum_k \mu(k)=\infty$ and $\kappa>0$, each simple-action is selected infinitely often. Also, if the averaging factor satisfies the two conditions $\sum_k \eta(k)=\infty$ and $\sum_k \left(\eta(k)\right)^2<\infty$, the estimated reward of each simple-action converges to the actual expected reward of the simple-action. Hence, as $\lambda(k)\to 0$ with $k\to\infty$, the selection probability of the most promising super-action converges to 1. Based on these observations, the proposed algorithm is ε-optimal.

3.3. Multi-reinforcement learning automata III

The multi-signal LA proposed in the previous section is not absolutely expedient. In order to illustrate this issue, consider the following scenario. A Multi-reinforcement LA II is operating in an environment, and its optimal super-action is $\underline{a}_i$. During the first k_1 updating steps, a super-action like $\underline{a}_j \neq \underline{a}_i$ receives more favorable feedbacks from the environment in comparison to the other super-actions. Hence, the following relation holds in step k_1. $\sum_l I_l Q_l(k_1) < \sum_l J_l Q_l(k_1)$, where I and J are Boolean vectors representing the members of $\underline{a}_i$ and $\underline{a}_j$, respectively. Accordingly, the selection probability of $\underline{a}_j$ would increase for some iterations after k_1 as $(\lambda(k))_{k\in\mathbb{N}}$ is a decreasing sequence of numbers. This increase in the selection probability of $\underline{a}_j$ continues, regardless of the received payoffs until the following condition is satisfied: $\sum_l I_l Q_l(k_2) > \sum_l J_l Q_l(k_2)$ for some k_2. Absolute expediency is a very useful property, which, under appropriate conditions, guarantees the convergence of a group of learning automata in a CLA structure [23, 26].

```
Algorithm II
Input
$r$: number of simple-actions
$A = \{a_1, a_2, \ldots, a_r\}$: actions set
$w<r$: number of simple-actions to be selected.
$(\lambda(k))_{k \in \mathbb{N}}$: learning rate.
$(\eta(k))_{k \in \mathbb{N}}$: averaging factor.

Initialization:
Initialize the estimated rewards for the simple-actions:
$Q_i(0) = c_1 \ \forall i \in \{1, \ldots, r\}$, where $c_1$ is some constant value.
repeat
    1. With probability $1-\lambda(k)$: // Greedy selection
        a. Select the $w$ simple-actions with the highest expected payoffs as $\alpha(k)$.
    2. Otherwise with probability $\lambda(k)/2$: // exploratory
        a. Select a random super-action as $\alpha(k)$.
    3. Otherwise: // mutated greedy
        a. Select the $w$ simple-actions with the highest expected payoffs as $\alpha(k)$.
        b. Select a random action $a_i \notin \alpha(k)$ according to its probability value defined as:
           $p_i = \frac{Q_i(k)}{\sum_{j \in I} Q_j(k)} \quad \forall i \in I = \{l \mid a_l \notin \alpha(k) \text{ and } a_l \in A\}$
        c. Select a random action $a_l \in \alpha(k)$ according to its probability value:
           $p_l = \frac{1-q_l}{\sum_{j \in I} 1-q_j} \quad \forall l \in I$
           where
           $q_h = \frac{Q_h(k)}{\sum_{j \in I} Q_j(k)} \quad \forall h \in I$
           and
           $I = \{s \mid a_s \in \alpha(k)\}$
        d. Replace $a_l$ with $a_i$ in $\alpha(k)$.
    4. Perform $\alpha(k)$ in the environment.
    5. Receive a vector reinforcement signal $\beta$.
    6. For each action $a_i$ in $\alpha(k)$ do:
        a. $Q_i(k+1) = Q_i(k) + \eta(k)(\beta_i(k) - Q_i(k))$
until some stopping condition is satisfied.
```

Figure 4. Pseudocode of multi-reinforcement learning automaton II.

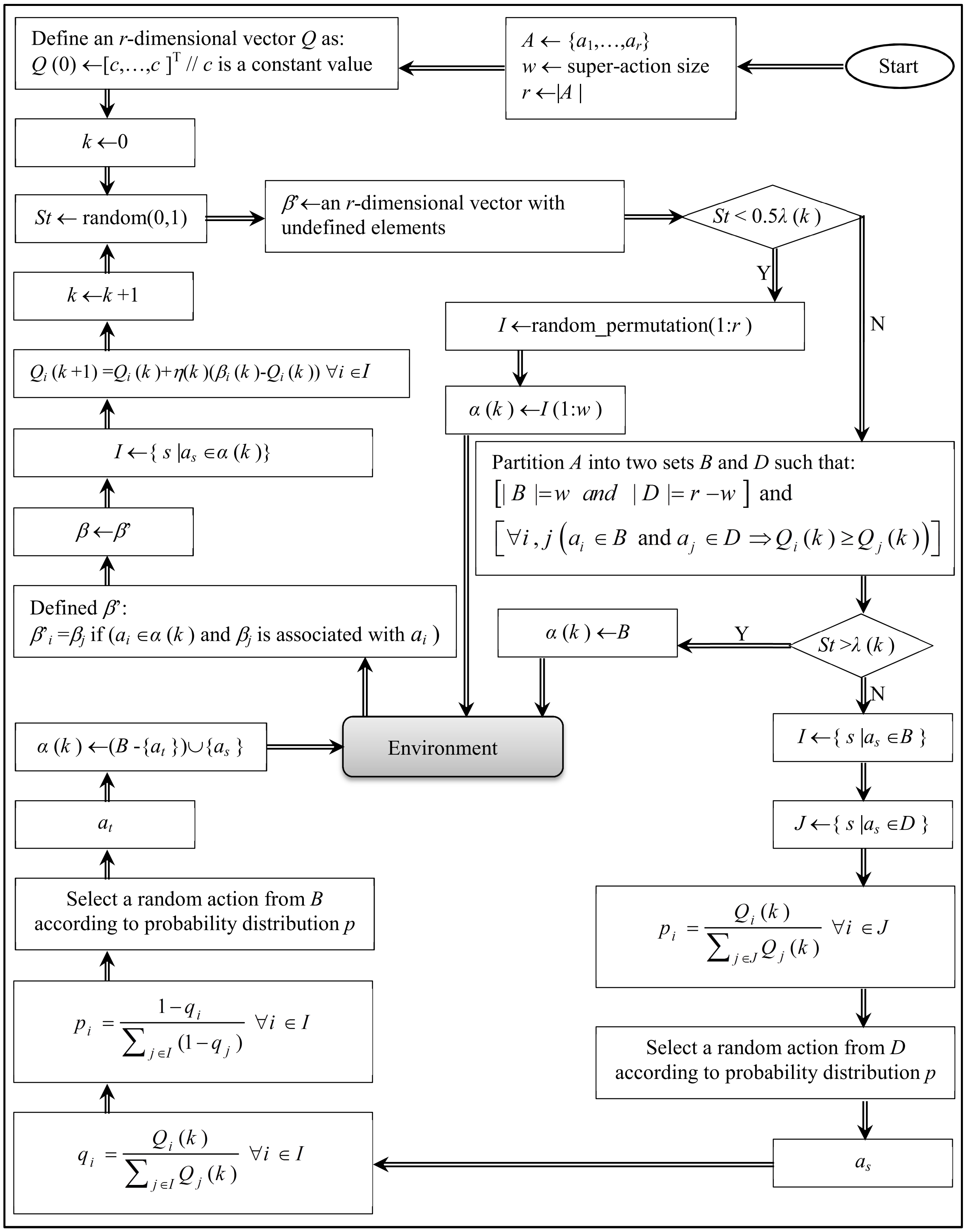

Start
$A \leftarrow \{a_1,\ldots,a_r\}$
$w \leftarrow$ super-action size
$r \leftarrow |A|$
Define an r-dimensional vector Q as:
$Q(0) \leftarrow [c,\ldots,c]^T$ // c is a constant value
$k \leftarrow 0$
$St \leftarrow$ random(0,1)
$\beta' \leftarrow$ an r-dimensional vector with undefined elements
$St < 0.5\lambda(k)$
Y
N
$I \leftarrow$ random_permutation(1:r)
$\alpha(k) \leftarrow I(1:w)$
Partition A into two sets B and D such that:
$\left[|B| = w \ \ and \ \ |D| = r - w\right]$ and
$\left[\forall i, j \left(a_i \in B \text{ and } a_j \in D \Rightarrow Q_i(k) \geq Q_j(k)\right)\right]$
$St > \lambda(k)$
Y
N
$\alpha(k) \leftarrow B$
$I \leftarrow \{s \mid a_s \in B\}$
$J \leftarrow \{s \mid a_s \in D\}$
$p_i = \frac{Q_i(k)}{\sum_{j \in J} Q_j(k)} \ \forall i \in J$
Select a random action from D according to probability distribution p
a_s
$q_i = \frac{Q_i(k)}{\sum_{j \in I} Q_j(k)} \ \forall i \in I$
$p_i = \frac{1 - q_i}{\sum_{j \in I}(1 - q_j)} \ \forall i \in I$
Select a random action from B according to probability distribution p
a_t
$\alpha(k) \leftarrow (B - \{a_t\}) \cup \{a_s\}$
Environment
Defined β':
$\beta'_i = \beta_j$ if ($a_i \in \alpha(k)$ and β_j is associated with a_i)
$\beta \leftarrow \beta'$
$I \leftarrow \{s \mid a_s \in \alpha(k)\}$
$Q_i(k+1) = Q_i(k) + \eta(k)(\beta_i(k) - Q_i(k)) \ \forall i \in I$
$k \leftarrow k + 1$

Figure 5. The block diagram of multi-reinforcement LA II

One of the absolutely expedient learning automata algorithms is L_{RI} [1]. In this section, another multi-reinforcement LA model is introduced which is based on the L_{RI} algorithm. Like algorithm I, each simple-action has an associated energy and is selected according to its energy level in each iteration. The internal state of the learning automaton is modeled as a particle in an r-dimensional hypercube. However, in contrast to algorithm I, the total energy of the LA (energy sum of all simple-actions) is always kept constant and equal to w. Accordingly, an increase in the energy level of a particular simple-action should results in a decrease in the energy levels of the others. In this regard, whenever a selected simple-action receives a favorable reinforcement, its energy level increases, and this increased amount is subtracted from the total energy level of the unselected simple-actions.

A multi-reinforcement learning automaton III operates as follows. At each iteration k, a w-dimensional super-action is selected in the same way as algorithm I. Then, this super-action is performed in the environment. The environment provides the learning system with a w-dimensional feedback signal, which represents the favorability of each one of the selected simple-actions in the super-action. If the associated signal to a selected simple-action like a_i is favorable, i.e. $\beta_i(k) = 1$, the energy level of a_i is increased according to Eq. (47). This increased amount is subtracted from the energy level of unselected actions in order to keep the total energy level of the LA constant (Eq. (48)).

$$x_i=x_i+\lambda\beta(k)(1-x_i) \qquad (47)$$

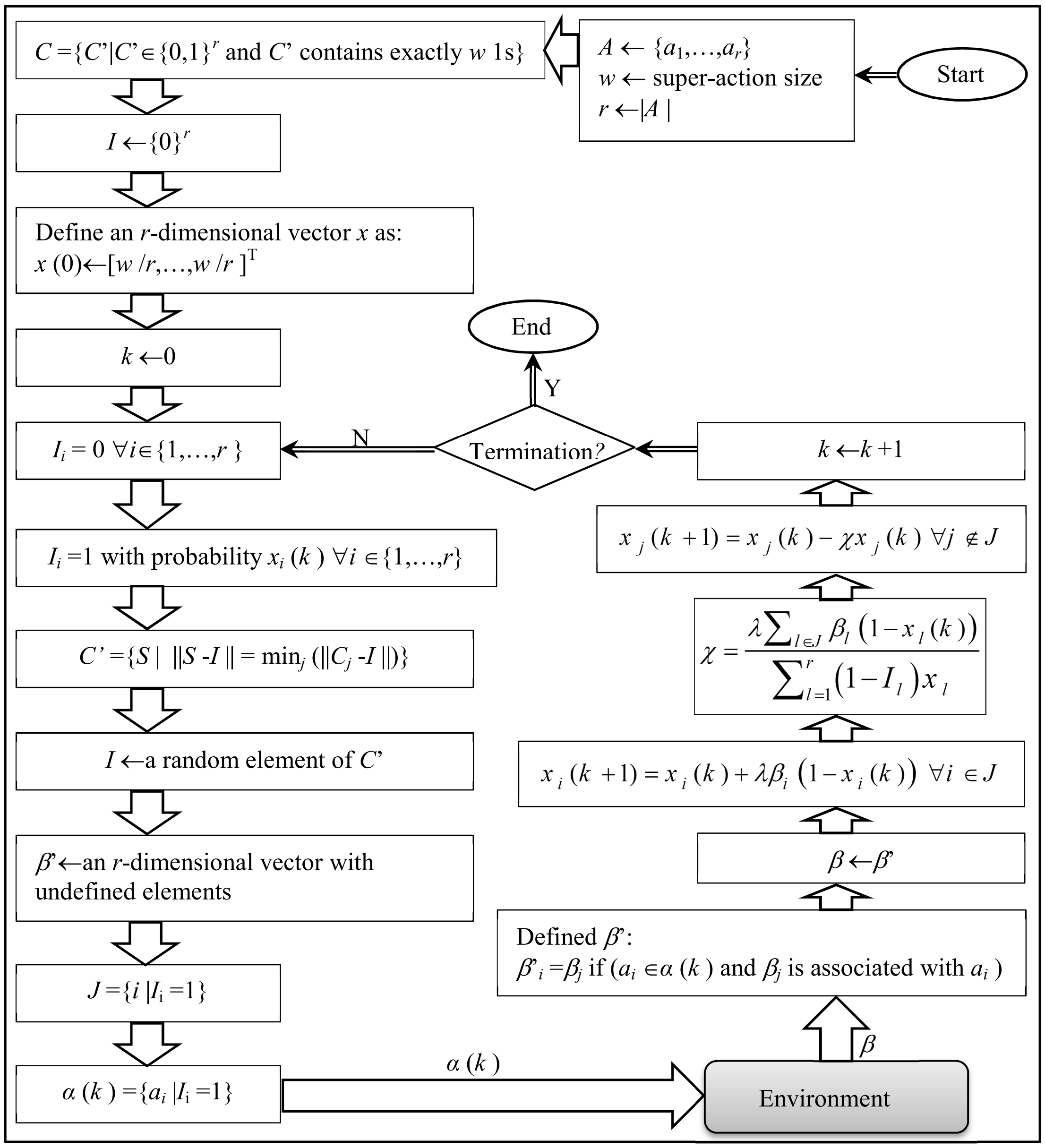
$$x_j = x_j - \frac{\lambda\beta_i(k)(1-x_i)}{\sum_l x_l(1-I_l)}(x_j) \ \forall j \in \{h \mid a_h \notin \alpha(k) \text{ and } a_h \in A\} \qquad (48)$$

Here, λ is the learning rate parameter, and similar to algorithm I, the selected set of simple-actions at step k is represented by $\alpha(k)$. I is an r-dimensional Boolean vector which represents the selected simple-actions. Fig. (6) illustrates the proposed multi-reinforcement learning automaton algorithm, and its block diagram is provided in Fig. (7). In implementation, each unselected action should be updated once for each rewarded action, (using a random order for the rewarded actions) as illustrated in Fig. 6. However, with a slight modification, we can first reward the selected actions, and then reduce the rewarded amount from the unselected actions as illustrated in Fig. 7. Both schemes yield very identical results and can be used alternatively.

4. Cellular learning automata models with multiple reinforcements

CLA is a hybrid model based on learning automata and cellular automata. In this model, each LA selects an action in each step of the learning process and performs the action in its environment. Then, the local rule of the CLA determines the feedback to the LA. This feedback is generated according to the selected actions in the neighborhood of the LA. The three LA models, introduced in section 3, can be directly integrated into the CLA model. The CLA models based on the three introduced multi-reinforcement LAs will be investigated empirically in the experimental section. In this section, we will investigate a CLA model which aims at maximizing the expected reward for each LA. It will be shown that the proposed model converges to a compatible point [43] when the local rules obey the characteristics of a potential function. Many distributed problems such as channel assignment in mesh networks can be modeled as potential games [48].

```
Algorithm III
Input
$r$: number of simple-actions
$A = \{a_1,\dots,a_r\}$
$w<r$: number of simple-actions to be selected.
$\lambda$: learning rate.
Initialization:
Set $x(0)=[w/r, w/r,\dots, w/r]^T$, where $x_i\in[0,1]$ represents the energy of the $i^{th}$ simple-action.
Consider an $r$-dimensional Boolean vector $I$: $I_i\in\{0,1\}$ $\forall i\in\{1,\dots,r\}$.
Let $C=\{C_1,C_2,\dots,C_o\}$ be the set of corners (vertices) of the LA hypercube with exactly $w$ 1s and ($r$-$w$) 0s,
where $o=|C|=\binom{r}{w}$, $C_{ij}\in\{0,1\}$, and $\sum_j C_{ij}=w \;\; \forall i\in\{1,\dots,o\}$.
Set $k = 0$.
Repeat
    1. Set $k= k+1$ and $\alpha(k)=\{\}$.
    2. Select a random set of simple-actions according to the energy level of the LA (i.e. according to
       $x$): each action like $a_i$ is selected with probability $x_i$.
    3. For each $i\in\{1,\dots,r\}$, set $I_i=1$ if action $a_i$ is chosen in step 2, otherwise set $I_i=0$.
    4. Let $C'$ be the subset of vertices in $C$, which are in the minimum distance to $I$. Randomly set $I$ to
       one of the vertices in $C'$.
    5. For each $i\in\{1,\dots,r\}$, add simple-action $a_i$ to $\alpha(k)$ if $I_i=1$.
    6. Apply $\alpha(k)$ to the environment.
    7. Receive an $r$-dimensional reinforcement signal $\beta$, where $\beta_j$ is undefined if $a_j\notin\alpha(k)$ (i.e. $I_j$=0).
    8. Update LA information as follows:
       For each action $a_i$ such that $I_i=1$ and $\beta_i(k)>0$ (in a random order) do:
           $x_j = x_j - \frac{\lambda\beta_i(k)(1-x_i)}{\sum_l x_l(1-I_l)}(x_j) \;\; \forall j\in\{h \mid a_h\notin\alpha(k) \text{ and } a_h\in A\}$
           $x_i=x_i+\lambda\beta(k)(1-x_i)$
until some stopping condition
```
Figure 6. Pseudocode of Multi-reinforcement learning automata III



$C=\{C'|C'\in\{0,1\}^r$ and C' contains exactly w 1s$\}$
$A\leftarrow\{a_1,\ldots,a_r\}$
$w\leftarrow$ super-action size
$r\leftarrow|A|$
Start
$I\leftarrow\{0\}^r$
Define an r-dimensional vector x as:
$x(0)\leftarrow[w/r,\ldots,w/r]^T$
End
$k\leftarrow 0$
Y
N
$I_i=0\ \forall i\in\{1,\ldots,r\}$
Termination?
$k\leftarrow k+1$
$I_i=1$ with probability $x_i(k)\ \forall i\in\{1,\ldots,r\}$
$x_j(k+1)=x_j(k)-\chi x_j(k)\ \forall j\notin J$
$C'=\{S\mid \|S-I\|=\min_j(\|C_j-I\|)\}$
$\chi=\frac{\lambda\sum_{l\in J}\beta_l(1-x_l(k))}{\sum_{l=1}^{r}(1-I_l)x_l}$
$I\leftarrow$a random element of C'
$x_i(k+1)=x_i(k)+\lambda\beta_i(1-x_i(k))\ \forall i\in J$
$\beta'\leftarrow$an r-dimensional vector with undefined elements
$\beta\leftarrow\beta'$
Defined β':
$\beta'_i=\beta_j$ if ($a_i\in\alpha(k)$ and β_j is associated with a_i)
$J=\{i|I_i=1\}$
β
$\alpha(k)$
$\alpha(k)=\{a_i|I_i=1\}$
Environment

Figure 7. The block diagram of multi-reinforcement LA III.

4.1. Multi-reinforcement cellular learning automata with the maximum expected rewards

In this section, a CLA model is introduced, which aims at maximizing the expected reward for each LA. The current CLA is closely related to the model previously proposed by Morshedlou and Meybodi [43]. In their model, each LA learns a probability distribution over its actions. This distribution leads to the maximum expected reward in response to the empirical joint distribution of actions that are performed by other learning automata in the neighborhood. Consequently, each LA converges to a probability distribution, which may have non zero probabilities on several actions. However, some applications require each LA to learn a single optimal action, or super-action. In this regard, the proposed multi reinforcement CLA, which will be called MCLA, estimates the payoff for each simple-action in response to the performed actions in the neighborhood. Then, it gradually increases the selection probability of the most favorable super-action, the one consisting of simple-actions with the highest expected rewards. The three introduced LA models can be generalized into CLA schemes which can maximize the expected reward of each LA according to distribution of actions in the neighborhood. For instance using LA-I model, each LA of the CLA selects a super-action as described in algorithm 1. Next, it observes the selected actions in its neighborhood and updates their empirical distributions. Then, the LA receives a feedback from the environment for each one of its selected simple-actions. Based on the empirical probability distributions and the received feedbacks, the estimated payoff of each combination of actions is updated. Finally similar to algorithm 1, the energies of simple-actions with the w_i highest expected rewards are increased. In what follows, a MCLA model based on LA-II will be investigated. In order to proceed, the required notations are summarized in Table 1. Additionally, some basic equations and relations can be derived as follows:

Table 1. Summary of basic notations and definitions used in MCLA.

Definition	Notation	Definition	Notation
Number of actions of the i^{th} LA	m_i	Action set of the i^{th} LA	$A_i = \{a_{i1},\ldots,a_{im_i}\}$
Number of actions to be selected by the i^{th} LA	w_i	Number of Super-actions of the i^{th} LA	$z_i = \binom{m_i}{w_i}$
Super-action set of the i^{th} LA	$\underline{A}_i = \{\underline{a}_{i1},\ldots,\underline{a}_{iz_i}\}$	A mixed strategy over $\underline{A}_i$ (a probability distribution over the elements $\underline{A}_i$)	σ
The set of all mixed strategies associated with $\underline{A}_i$	Δ_i	Index of cells in the neighborhood of the cell i	$N_i(1), N_i(2), \ldots, N_i(n_i)$
The number of neighbors of the i^{th} cell (LA)	n_i	Average reinforcement of a simple-action like a_{ij} for the neighbors' joint actions until step k	$Q_i(k, a_{ij}, \underline{a}_{N_i(1)l_1}, \ldots, \underline{a}_{N_i(n_i)l_{n_i}})$
Estimated selection probability of $\underline{a}_{ij}$ in step k	$\hat{p}_{ij}(k)$		

Empirical action selection probability of the i^{th} LA in step k: $\hat{p}_i(k) = \left[\hat{p}_{i1}(k), \hat{p}_{i2}(k), \ldots, \hat{p}_{iz_i}(k)\right]^{\mathrm{T}}$

Estimated action selection probabilities in the neighborhood of cell i: $\hat{p}n_i(k) = \left[\hat{p}_{N_i(1)}(k), \ldots, \hat{p}_{N_i(n_i)}(k)\right]$

Expected reward for an action like a_{ij} at time step k:

$$Er(a_{ij},\hat{p}n_i(k))= \sum_{l_1=1}^{z_{N_i(1)}}\dots\sum_{l_{n_i}=1}^{z_{N_i(n_i)}}\left(\prod_{s=1}^{n_i}\hat{p}_{N_i(s)l_s}(k)\right)Q_i(k,a_{ij},\underline{a}_{N_i(1)l_1},\dots,\underline{a}_{N_i(n_i)l_{n_i}}) \tag{49}$$

Super-action set with the maximum expected reward for the ith LA at time step k:

$$B_i(k,\hat{p}n_i(k))=\left\{\underline{a}_{ih}\in\underline{A}_i \mid \sum_{a_{il}\in\underline{a}_{ih}}Er(a_{il},\hat{p}n_i(k))=\max_{\underline{a}_{ij}\in\underline{A}_i}\left(\sum_{a_{il}\in\underline{a}_{ij}}Er(a_{il},\hat{p}n_i(k))\right)\right\} \tag{50}$$

Actual reward for the mixed strategy σ:

$$F_i(\sigma,\hat{p}n_i(k))= \sum_{u=1}^{z_i}\sigma_u\sum_{a_{ij}\in\underline{a}_{iu}}\left[\sum_{l_1=1}^{z_{N_i(1)}}\dots\sum_{l_{n_i}=1}^{z_{N_i(n_i)}}\left(\prod_{s=1}^{n_i}\hat{p}_{N_i(s)l_s}(k)\right)F_i(k,a_{ij},\underline{a}_{N_i(1)l_1},\dots,\underline{a}_{N_i(n_i)l_{n_i}})\right] \tag{51}$$

Epsilon maximum expected reward set for the ith LA:

$$b_i^{\varepsilon}(k,\hat{p}n_i(k))=\left\{\sigma\in\Delta_i \mid \max_{\sigma'\in\Delta_i}\left(F_i(\sigma',\hat{p}n_i(k))\right)-F_i(\sigma,\hat{p}n_i(k))\le\varepsilon\right\} \tag{52}$$

```
Algorithm IV
Input
$r$: number of actions
$w<r$: number of actions to be selected.
$(\lambda(k))_{k\in\mathbb{N}}$: learning rate.
$(\eta(k))_{k\in\mathbb{N}}$: averaging factor.
$(\gamma(k))_{k\in\mathbb{N}}$: discount factor.

While termination condition is not satisfied, do:
I. For each cell like $i$, synchronously with other cells, do:
    1. With probability $\lambda(k)/2$ do: // exploratory selection
        1. Select a random super-action as $\alpha_i(k)$.
    2. With probability $\lambda(k)/2$ do: // mutated greedy selection
        1. Select the maximum expected reward super-action let $\alpha_i(k)$ be this action.
        2. Define $I$ as $I=\{l|a_{il}\in\alpha_i(k)\}$
        3. Select a random simple-action from $\alpha_i(k)$ according to its probability value
           defined as:
           $p_l=\frac{1-q_l(k)}{\sum_{j\in I}1-q_j(k)}\quad\forall l\in I$ ,
           where
           $q_h=\frac{Er(a_{ih},\hat{p}n_i(k))}{\sum_{a_{il}\in\alpha_i(k)}Er(a_{il},\hat{p}n_i(k))}\quad\forall h\in I$
        4. Select a random simple-action not in $\alpha_i(k)$ according to the probability values
           defined as:
                $p_h = \frac{Er\left(a_{ih}, \hat{p}n_i(k)\right)}{\sum_{a_{is} \notin \alpha_i(k)} Er\left(a_{is}, \hat{p}n_i(k)\right)} \quad \forall h \in \{l \mid a_{il} \in A_i \;\&\; a_{il} \notin \alpha_i(k)\}$
            5.  Replace the action obtained in step 2.3 with the one obtained in step 2.4 in $\alpha_i(k)$.
    3.  With probability 1- $\lambda(k)$ do:
            1.  Select the maximum expected reward super-action as $\alpha_i(k)$.
    4.  Perform the selected super-action, and receive the reinforcement vector $\beta_i(k)$.
    5.  $\hat{p}_{ij}(k+1) = \begin{cases} (1-\gamma(k))\hat{p}_{ij}(k) + \gamma(k) & \text{if } \alpha_i(k) = a_{ij} \\ (1-\gamma(k))\hat{p}_{ij}(k) & \text{otherwise} \end{cases} \quad \forall j=1,\ldots,z_i$
    6.  $Q_i(k+1,:)=Q_i(k,:)$
        $Q_i(k+1, a_{ij}, \alpha_{N_i(1)}(k), \ldots, \alpha_{N_i(n_i)}(k)) =$
    7.  $Q_i(k, a_{ij}, \alpha_{N_i(1)}(k), \ldots, \alpha_{N_i(n_i)}(k)) +$          $\forall j \in \{l \mid a_{il} \in \alpha_i(k)\}$
        $\eta(k)\left(\beta_{ij}(k) - Q_i(k, a_{ij}, \alpha_{N_i(1)}(k), \ldots, \alpha_{N_i(n_i)}(k))\right)$
End for
II. Set $k \leftarrow k+1$.
End while
```

Figure 8. the pseudocode of MCLA

Fig. (8) gives a pseudocode for the described MCLA model, and the block diagram of the model is provided in Fig. (9). In this model, each learning automaton learns the empirical distribution of actions that are occurring in its neighborhood. Also, it learns the estimated payoff of each one of its simple-actions in combination with different super-actions, which are performed by the neighboring learning automata. Consequently, it can compute the expected reward for each one of its simple-actions. During each iteration k, every LA like LA_i finds its w_i simple-actions with the highest expected rewards. Then, with probability 1-$\lambda(k)$, the LA performs these actions in the environment, or with probability $\lambda(k)$ it experiments two other action selection strategies similar to algorithm II. After performing these selected simple-actions, the LA updates the empirical distribution of the actions of neighboring learning automata and the estimated payoff of its performed actions. It should be noted that $\lambda(k)$ is a decreasing sequence of numbers; accordingly, the selection probability of the most promising super-actions approaches to one during the learning procedure.

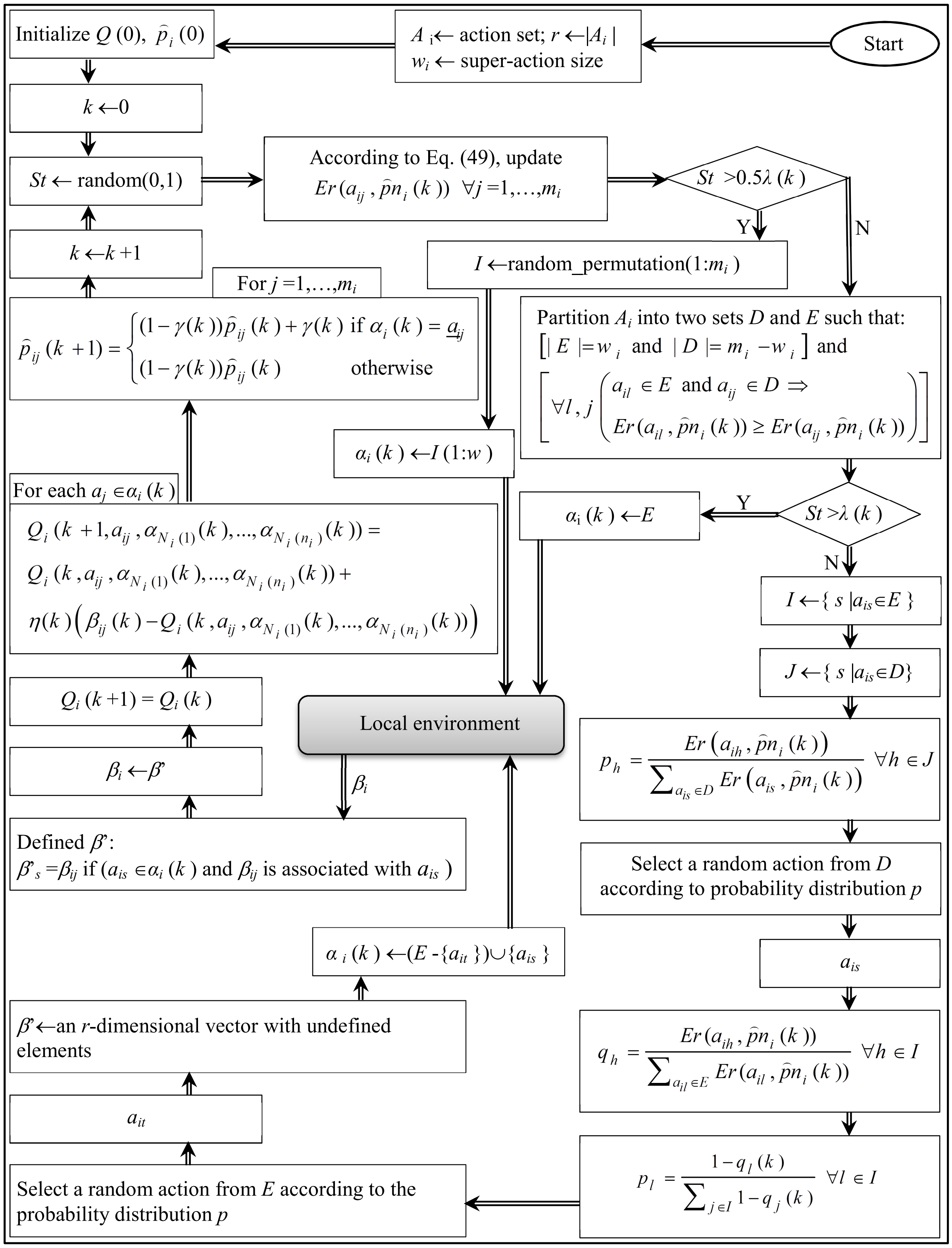

Start
$A_i \leftarrow$ action set; $r \leftarrow |A_i|$
$w_i \leftarrow$ super-action size
Initialize $Q(0)$, $\hat{p}_i(0)$
$k \leftarrow 0$
$St \leftarrow$ random(0,1)
According to Eq. (49), update $Er(a_{ij}, \hat{pn}_i(k))$ $\forall j = 1, \ldots, m_i$
$St > 0.5\lambda(k)$
Y
N
$I \leftarrow$ random_permutation($1{:}m_i$)
$\alpha_i(k) \leftarrow I(1{:}w)$
Partition A_i into two sets D and E such that: $[|E| = w_i$ and $|D| = m_i - w_i]$ and $\left[\forall l, j \left(\begin{array}{l} a_{il} \in E \text{ and } a_{ij} \in D \Rightarrow \\ Er(a_{il}, \hat{pn}_i(k)) \geq Er(a_{ij}, \hat{pn}_i(k)) \end{array}\right)\right]$
$St > \lambda(k)$
Y
N
$\alpha_i(k) \leftarrow E$
$I \leftarrow \{ s \mid a_{is} \in E \}$
$J \leftarrow \{ s \mid a_{is} \in D \}$
$p_h = \frac{Er(a_{ih}, \hat{pn}_i(k))}{\sum_{a_{is} \in D} Er(a_{is}, \hat{pn}_i(k))}$ $\forall h \in J$
Select a random action from D according to probability distribution p
a_{is}
$q_h = \frac{Er(a_{ih}, \hat{pn}_i(k))}{\sum_{a_{il} \in E} Er(a_{il}, \hat{pn}_i(k))}$ $\forall h \in I$
$p_l = \frac{1 - q_l(k)}{\sum_{j \in I} 1 - q_j(k)}$ $\forall l \in I$
Select a random action from E according to the probability distribution p
a_{it}
$\beta' \leftarrow$ an r-dimensional vector with undefined elements
$\alpha_i(k) \leftarrow (E - \{a_{it}\}) \cup \{a_{is}\}$
Local environment
β_i
Defined β': $\beta'_s = \beta_{ij}$ if ($a_{is} \in \alpha_i(k)$ and β_{ij} is associated with a_{is})
$\beta_i \leftarrow \beta'$
$Q_i(k+1) = Q_i(k)$
For each $a_j \in \alpha_i(k)$
$Q_i(k+1, a_{ij}, \alpha_{N_i(1)}(k), \ldots, \alpha_{N_i(n_i)}(k)) = Q_i(k, a_{ij}, \alpha_{N_i(1)}(k), \ldots, \alpha_{N_i(n_i)}(k)) + \eta(k)\left(\beta_{ij}(k) - Q_i(k, a_{ij}, \alpha_{N_i(1)}(k), \ldots, \alpha_{N_i(n_i)}(k))\right)$
For $j = 1, \ldots, m_i$
$\hat{p}_{ij}(k+1) = \begin{cases} (1-\gamma(k))\hat{p}_{ij}(k) + \gamma(k) & \text{if } \alpha_i(k) = a_{ij} \\ (1-\gamma(k))\hat{p}_{ij}(k) & \text{otherwise} \end{cases}$
$k \leftarrow k + 1$

Figure 9. A block diagram for the operation of a typical LA like LA_i in the MCLA. All LA in the MCLA run the similar code and operate synchronously.

4.2 Convergence analysis of MCLA

Lemma 3. Assume that the following hold w.p.1: $E\left[F_i(a_{ij},a')\right]$ = A, $E\left[F_i(a_{ij},a')^2\right]<\infty$, $\sum_{k=1}^{\infty}\eta(k)=\infty$, and $\sum_{k=1}^{\infty}\eta(k)^2<\infty$. Also, assume that $Q_i(.,a_{ij},a')$ is updated infinitely often. Then, $Q_i(k,a_{ij},a')$ converges to A w.p.1 as $k\to\infty$ [10].

Lemma 4. If $\lambda(k)=2\tau rk^{-1/s}$, where r is maximum number of super-actions in the LAs of the MCLA and s is the maximum neighborhood size in the MCLA, assuming the conditions of lemma 3 is held, then in the proposed model:

$$\lim_{k\to\infty}\left|Q_i(k,a_{ij},a')-F_i(a_{ij},a')\right|=0 \quad \forall i,j,a' \quad (53)$$

Proof. For a typical LA like LA$_i$, the selection probability of each super-action is at least $\dfrac{\lambda(k)}{2\underline{m}_i}\geq\dfrac{\lambda(k)}{2r}$. Accordingly, the occurrence of each joined super-action in the neighborhood of LA$_i$ is at least $\left(\dfrac{\lambda(k)}{2r}\right)^{n_i}\geq\left(\dfrac{\lambda(k)}{2r}\right)^{s}$. With $\lambda(k)=\tau 2rk^{-1/s}$, the selection probability of each joint action is at least $\tau^s k^{-1}$. $\tau^s>0$ is constant, and it can be verified that $\sum_{k=1}^{\infty}k^{-1}=\infty$. Accordingly, the average estimated reinforcements converge to the actual ones w.p.1 [10].

Theorem 2. A MCLA with potential local rules converges to a set of Nash equilibria under the following conditions.

We assume that the conditions of lemma 3 and lemma 4 are satisfied and, also, assume that $(\gamma(k))_{k\in\mathbb{N}}$ is a sequence satisfying the two requirements $\sum_{k=1}^{\infty}\gamma(k)=\infty$ and $\sum_{k=1}^{\infty}\left(\gamma(k)\right)<\infty$. Additionally, the reinforcement signals are bounded and fall in the range $[0,D_{\max}]$.

Following lemma 4, $\lim_{k\to\infty}\left|Q_i(k,a_{ij},a')-F_i(a_{ij},a')\right|=0 \ \forall i,j,a'$, and consequently, there is a sequence like $(\mu_i(k))_{k\in\mathbb{N}}$ such that $\mu_i(k)\to 0$ as $k\to\infty$ and

$$\left|Q_i(k,a_{ij},a')-F_i(a_{ij},a')\right|<\mu_i(k) \quad \forall i,j,a' \ . (54)$$

We first show that, at each iteration k, the super-action of a typical LA like LA_i is selected from the set of its epsilon maximum expected payoff super-action set with respect to $\hat{pn}_i(k)$. To do this, it should be shown that the expected reward of the selected actions of LA_i plus ε is greater than the maximum of the achievable actual reward.

During each time step k, the action of a learning automaton like LA_i is selected according to a mixed strategy like $\pi_i(k)$ (considering three action selection strategies in Fig. (8)). Also, by representing the expected reward of the i^{th} cell at time k with $Er_i(k)$ and following the procedure of multi-signal cellular learning automata in Fig. (8), we have

$$\begin{aligned}
&Er_i(k) \geq \left(1-\lambda(k)\right) \max_{\underline{a}_{ij} \in \underline{A}_i} \left(\sum_{a_{il} \in \underline{a}_{ij}} Er(a_{il}, \hat{pn}_i(k)) \right) + \\
&\frac{\lambda(k)}{2} \frac{1}{\left|\underline{A}_i\right|} \sum_{\underline{a}_{ij} \in \underline{A}_i} \sum_{a_{il} \in \underline{a}_{ij}} Er(a_{il}, \hat{pn}_i(k)) + \\
&\frac{\lambda(k)}{2} \left[\sum_{\underline{a}_{ij} \in B_i(k)} \sum_{a_{il} \in \underline{a}_{ij}} (1-\frac{1}{w_i}) Er(a_{il}, \hat{pn}_i(k)) + \frac{1}{m_i - w_i} \sum_{a_{ih} \notin \underline{a}_{ij}} Er(a_{ih}, \hat{pn}_i(k)) \right]
\end{aligned} \tag{55}$$

$$\begin{aligned}
&Er_i(k) \geq \left(1-\lambda(k)\right) \max_{\underline{a}_{ij} \in \underline{A}_i} \left(\sum_{a_{il} \in \underline{a}_{ij}} Er(a_{il}, \hat{pn}_i(k)) \right) \geq \\
&\left(1-\lambda(k)\right) \left[\max_{\underline{a}_{ij} \in \underline{A}_i} \left(\sum_{a_{il} \in \underline{a}_{ij}} F(a_{il}, \hat{pn}_i(k)) \right) - w_i \mu_i \right] \geq \\
&\max_{\underline{a}_{ij} \in \underline{A}_i} \left(\sum_{a_{il} \in \underline{a}_{ij}} F(a_{il}, \hat{pn}_i(k)) \right) - \lambda(k) w_i D_{\max} - w_i \mu_i + \lambda(k) w_i \mu_i = \\
&\max_{\underline{a}_{ij} \in \underline{A}_i} \left(\sum_{a_{il} \in \underline{a}_{ij}} F(a_{il}, \hat{pn}_i(k)) \right) + \varepsilon_i(k)
\end{aligned} \tag{56}$$

As $k \to \infty$, we have $\mu_i \to 0$ and $\lambda(k) \to 0$; consequently, $\varepsilon_i(k) \to 0$. Therefore, the selected super-action of each LA converges to its epsilon maximum expected payoff super-action set.

Based on the updating rules of the proposed algorithm, $\hat{p}_i(k+1) = (1-\gamma_{k+1})\hat{p}_i(k) + \gamma_{k+1} I_{\alpha_i(k)}$. The expected value of $I_{\alpha_i(k)}$ is $\pi_i(k)$. Consequently, $I_{\alpha_i(k)} = \pi_i(k) + M_i(k)$, where $M_i(k)$ is the difference between the expected value $I_{\alpha_i(k)}$ at step k and its actual value. As discussed before

$\pi_i(k) \in b_i^{\varepsilon}(k, \hat{p}n_i(k))$. Accordingly, $\hat{p}_i(k+1) \in (1-\gamma_{k+1})\hat{p}_i(k) + \gamma_{k+1}\left(b_i^{\varepsilon}(\hat{p}n_i(k)) + M_i(k)\right)$. $M_i(k)$ is a sequence of perturbations such that, with the assumed conditions on $\gamma(k)$,

$$\lim_{k\to\infty} \sup_t \left\{ \left\| \sum_{j=k}^{t-1} \gamma_i(j+1) M_i(j+1) \right\| \mid \sum_{j=k}^{t-1} \gamma_i(j+1) \le T \right\} = 0 \quad \text{for any } T>0 \quad (57)$$

Accordingly, $\hat{p}_i(k+1) \in (1-\gamma_{k+1})\hat{p}_i(k) + \gamma_{k+1}\left(b_i^{\varepsilon}(\hat{p}n_i(k)) + M_i(k)\right)$ is a GWFP process [49], and converges to the set of Nash equilibria if local rules satisfy the conditions in Eq. (10).

4.3 On the computational complexity of MCLA

The proposed MCLA has updating steps with exponential computational complexities, which can significantly affect its performance. However, considering the properties of local rules in some applications, the time complexity of the approach can be reduced effectively. The multi-reinforcement RL schemes introduced in this paper are developed on the basis that the environment can provide each selected simple-action of a super-action with an individual reinforcement signal. Accordingly, different simple-actions of a super-action are learned separately, which significantly reduces the learning time. The same approach can be employed in MCLA. For instance, consider a multi agent environment where each agent interacts with several other agents in its neighborhood. During each iteration, a typical agent selects a set of actions and applies each action to one of its neighbors. The neighboring agent which receives an action responds the executer of the action based on its own selected actions. Accordingly, an LA can learn the estimated payoff for each simple-action in combination with different other simple-actions rather than joined super-actions. Then, the expected rewards for simple-actions can be calculated based on these approximated payoffs. In the local rule of the application which is studied in the next section, the expected payoff for each simple-action only depends on the selection of the same simple-action by the neighboring LAs. Hence, its learning procedure can be implemented in an efficient manner. Additionally, we can simply use the RL algorithms, proposed in section 3, in each cell of the CLA. All of these algorithms have linear time updating steps, and as it will be demonstrated in the experimental section, they can achieve very promising solutions.

5. Case study: distributed channel assignment in multi-radio wireless networks

The proposed LA and CLA models can be employed for solving several engineering and scientific problems. As an application of the proposed models, this section examines the multi-reinforcement CLA models on a channel assignment problem. The proposed methods can be applied to different channel assignment tasks such as the one described in [50].

5.1. System model

In this study, several pairs of users are communicating with each other over single hops. Each pair of such users will be modeled as a node in the system. Each user possesses a set of identical radios that can be tuned on the available channels. It is assumed that the available frequency band is divided into a set of orthogonal channels with identical bandwidths. The communication between two users in the same node is bidirectional, and the two participants in a communication can coordinate to select identical channels. As the users can possess several radios, they can communicate with each other at the same time over multiple channels. The available channels for node i in the system will be represented by S_i, and the number of its equipped radios will be denoted by K_i with $K_i \leq |S_i|$. The communication between two users in a node over a channel may be unsuccessful due to interference from other nodes. In this paper, the interference model described in [51] will be employed: if two nodes within each other's interference range transmit data on a same channel, none of them can transmit any data successfully. We also assume that the users in different nodes do not cooperate with each other, and each one tries selfishly to maximize its own throughput. The throughput of each user is maximized if it can select the maximum number of noninterfering channels. Additionally, a user does not use a similar channel on more than one radio to avoid interference from itself. Nodes may leave some of their radios idle if they cannot find any noninterfering channels for these radios. A node can be encouraged to leave some of its radios idle and not to assign them interfering channels by getting small rewards on its idle radios. Idle radios on a node would provide chances for some other nodes to tune their radios on more channels.

Possible interferences between different nodes of a network can be modeled using a conflict graph. Each pair of users in the system is represented by a node in the conflict graph. There is a link between two nodes in the conflict graph if and only if the two nodes are in the interference range of each other. Hence, a user can transmit successfully on a particular channel if none of the users in its adjacent nodes in the conflict graph transmit on that channel. An example conflict graph of a network with 25 nodes is depicted in Fig. (10).

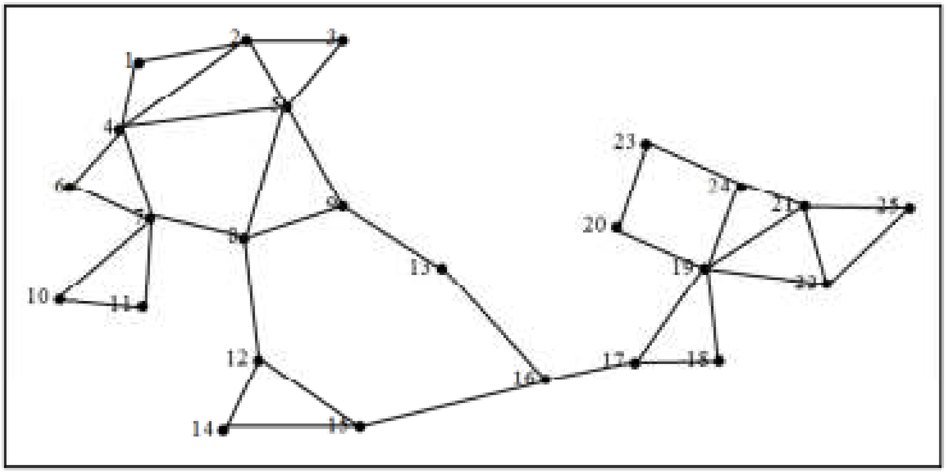


Figure 10. A sample conflict graph of a wireless network with 25 pairs of users. Each pair of users is modeled as a node in the conflict graph, and the possible interferences between nodes are represented by the edges of the graph.

5.2. Channel assignment algorithm based on CLA

In order to solve the introduced channel assignment problem by the CLA approach, a CLA with N cells is considered where N represents the number of nodes in the network. The CLA is isomorphic with the considered network, i.e. cells i and j are connected if and only if nodes i and j are adjacent in the network. Each cell contains an LA which controls the channel assignment strategy of its associated node.

Each LA, like LA_i, has an action set of size $|S_i| + K_i$, where K_i represents the number of equipped radios on each user in the i^{th} node, and S_i represents the available channels of the node. Here, the action set size is $|S_i| + K_i$ rather than $|S_i|$ since each node can choose to leave some of its radios in an idle state. During each iteration of the assignment procedure, each LA, like LA_i, selects a set of K_i simple-actions according to its internal state. Then, these chosen simple-actions are assigned to the radios of the associated nodes. After action selection step, each LA receives a reinforcement signal which represents the favorability of its selected simple-actions. In the proposed algorithm, each selected simple-action corresponding to an available channel receives one reward point if the channel is not selected in any of the neighboring nodes (or cells). If a chosen channel is also selected in a neighboring node, the corresponding simple-action receives no reward. A simple-action corresponding to an idle radio always receives $\varepsilon<1$ reward point (or receives one reward point with probability ε). Accordingly, nodes are encouraged not to select interfering channels so that these channels can be utilized by the nearby nodes. The CLA of the proposed channel assignment algorithm can be implemented based on any one of the LA models that are described in section 3 or can be implemented based on the model described in section 4.

Fig. (11) illustrates the channel assignment algorithm based on the CLA which uses LA III. In order to illustrate the operation of the channel assignment algorithm, the procedure in Fig. (11) is presented in sequential form. However, each node of the network runs its code synchronously with other nodes of the network as illustrated in Fig. (12). The CLA in Fig. (12) is based on LA II. Algorithms based on LA I and MCLA are much similar to the ones illustrated in Fig. (12); hence, they are not provided here for the sake of space economy.

```
Input:
$G = (V, E)$: conflict graph of the network
$S = (S_1, S_2, \ldots, S_N)$. $S_i$: the set of available Channels for node $i$ in the network.
$K = (K_1, K_2, \ldots, K_N)$. $K_i$: number of equipped radios in node $i$.
$\lambda$: learning rate of each LA.
Initialization:
Consider an irregular CLA isomorphic with the network.
$LA_i$ in the CLA has the following characteristics:
        It is a multi-signal LA with $r_i = |S_i| + K_i$ simple-actions and super-action size of $K_i$.
        $A_i = \{a_{i1}, \ldots, a_{ir_i}\}$.
        The energy level of the LA is represented by an $r_i$-dimensional vector $x_i$, where
        $x_{ij} \in [0,1]$ represents the energy level of the $j^{th}$ simple-action
        $x_i(0) = [K_i/r_i, K_i/r_i, \ldots, K_i/r_i]^T$.
Set $t \leftarrow 0$.
Repeat
    1. For each cell like cell $i$ do:
        a. Select a set of simple-actions according to their energy levels; name it $\alpha'_i(t)$.
        b. Randomly select one of the closest super-actions to $\alpha'_i(t)$; denote it by $\alpha_i(t)$.
    2. Generate a reinforcement vector for each LA like $LA_i$:
        a. For each action $a_{ij} \in \alpha_i(t)$ do:
            i. If $a_{ij} \in S_i$:
                $\beta_{ij}(t) = \begin{cases} 1 \text{ if } a_{ij} \text{ is not selected in any neighboring node} \\ 0 \text{ otherwise} \end{cases}$
            ii. If $a_{ij} \notin S_i$:
                $\beta_{ij}(t) = \begin{cases} 1 \text{ with probability } \varepsilon \\ 0 \text{ otherwise} \end{cases}$
    3. Update each LA like $LA_i$ as follows:
        a. Define $J = \{j \mid a_{ij} \in \alpha_i(t)\}$.
        b. Define $I = \{j \mid a_{ij} \in A_i - \alpha_i(t)\}$.
        c. Update the energy level of the LA as follows:
            $x_{ij}(t+1) = x_{ij}(t) + \lambda\beta_{ij}(t)(1 - x_{ij}(t)) \quad \forall j \in J$
            $\chi = \frac{\lambda \sum_{l \in J} \beta_{il}(1 - x_{il}(t))}{\sum_{l \in I} x_{il}(t)}$
            $x_{ij}(t+1) = x_{ij}(t) - \chi x_{ij}(t) \quad \forall j \notin J$
    4. Set $t \leftarrow t+1$
until some stopping condition is satisfied
```

Figure 11. Channel assignment procedure based on CLA with multi-signal LA type III

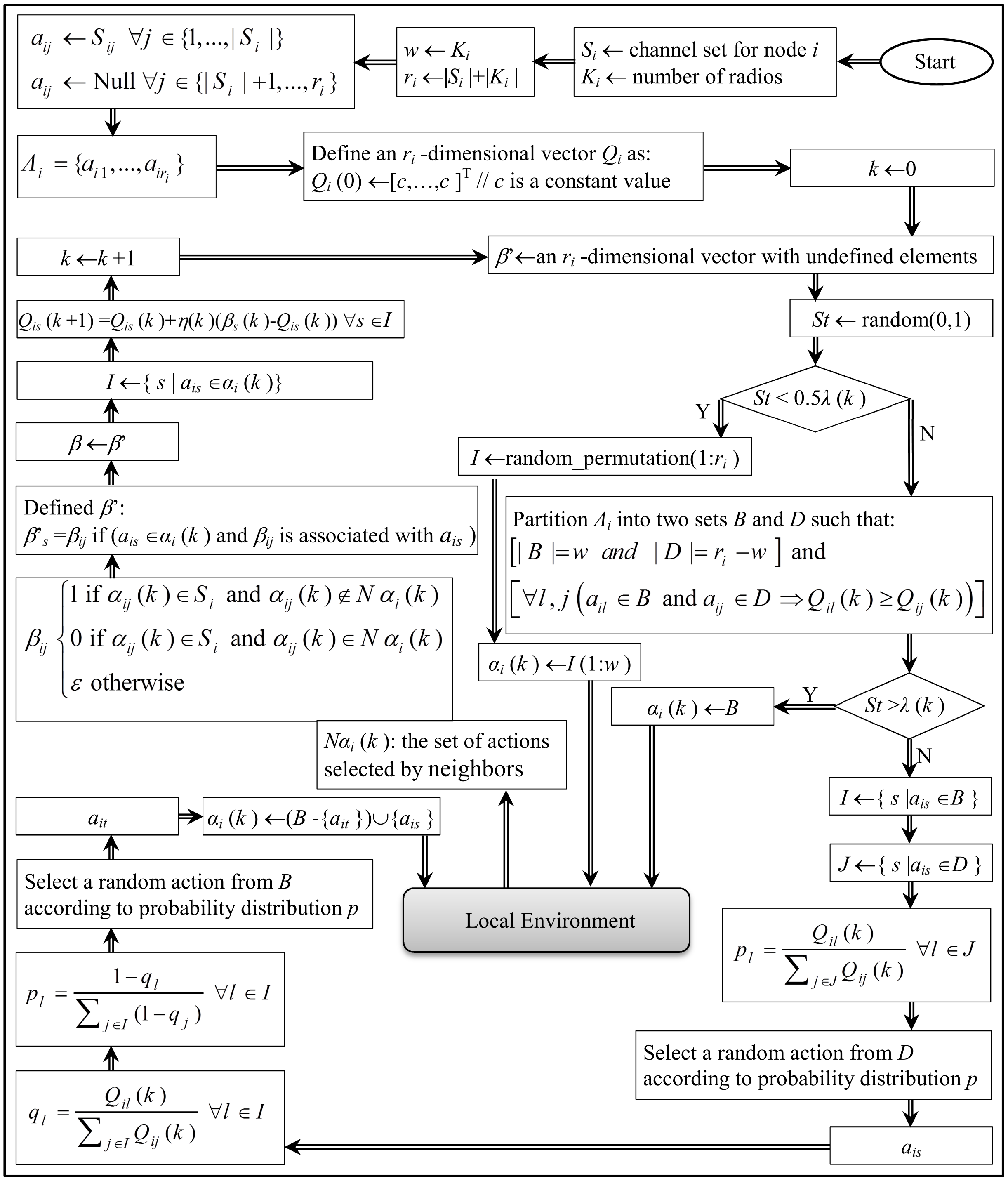

Start
$S_i \leftarrow$ channel set for node i
$K_i \leftarrow$ number of radios
$w \leftarrow K_i$
$r_i \leftarrow |S_i| + |K_i|$
$a_{ij} \leftarrow S_{ij} \ \forall j \in \{1,\dots,|S_i|\}$
$a_{ij} \leftarrow$ Null $\forall j \in \{|S_i|+1,\dots,r_i\}$
$A_i = \{a_{i1},\dots,a_{ir_i}\}$
Define an r_i-dimensional vector Q_i as:
$Q_i(0) \leftarrow [c,\dots,c]^T$ // c is a constant value
$k \leftarrow 0$
$\beta' \leftarrow$ an r_i-dimensional vector with undefined elements
$St \leftarrow$ random(0,1)
$St < 0.5\lambda(k)$
Y
N
$I \leftarrow$ random_permutation($1{:}r_i$)
Partition A_i into two sets B and D such that:
$\left[|B| = w \ \ and \ \ |D| = r_i - w\right]$ and
$\left[\forall l, j \left(a_{il} \in B \text{ and } a_{ij} \in D \Rightarrow Q_{il}(k) \geq Q_{ij}(k)\right)\right]$
$\alpha_i(k) \leftarrow I(1{:}w)$
$St > \lambda(k)$
Y
N
$\alpha_i(k) \leftarrow B$
$I \leftarrow \{s \mid a_{is} \in B\}$
$J \leftarrow \{s \mid a_{is} \in D\}$
$p_l = \frac{Q_{il}(k)}{\sum_{j \in J} Q_{ij}(k)} \ \forall l \in J$
Select a random action from D according to probability distribution p
a_{is}
$q_l = \frac{Q_{il}(k)}{\sum_{j \in I} Q_{ij}(k)} \ \forall l \in I$
$p_l = \frac{1 - q_l}{\sum_{j \in I}(1 - q_j)} \ \forall l \in I$
Select a random action from B according to probability distribution p
a_{it}
$\alpha_i(k) \leftarrow (B - \{a_{it}\}) \cup \{a_{is}\}$
Local Environment
$N\alpha_i(k)$: the set of actions selected by neighbors
$\beta_{ij} \begin{cases} 1 \text{ if } \alpha_{ij}(k) \in S_i \text{ and } \alpha_{ij}(k) \notin N\alpha_i(k) \\ 0 \text{ if } \alpha_{ij}(k) \in S_i \text{ and } \alpha_{ij}(k) \in N\alpha_i(k) \\ \varepsilon \text{ otherwise} \end{cases}$
Defined β':
$\beta'_s = \beta_{ij}$ if ($a_{is} \in \alpha_i(k)$ and β_{ij} is associated with a_{is})
$\beta \leftarrow \beta'$
$I \leftarrow \{s \mid a_{is} \in \alpha_i(k)\}$
$Q_{is}(k+1) = Q_{is}(k) + \eta(k)(\beta_s(k) - Q_{is}(k)) \ \forall s \in I$
$k \leftarrow k+1$

Figure 12. Block diagram for the operation of the i^{th} cell of the CLA based on LA-II. The operation of the i^{th} cell is performed in node i of the network. All nodes run similar codes in synchronous manner.

6. Experimental studies

In this section, we experimentally analyze the performance and the convergence behavior of the proposed multi-reinforcement learning schemes. In this regard, we first illustrate the applicability of the proposed multi-reinforcement LA schemes on a set of decision making problems. Then, the performance of all CLA models on the channel assignment problem given in section 5 will be examined.

6.1 Performance evaluation of the multi reinforcement LA models

This section illustrates the performance of the proposed LA models. We assume that there is a hypothetical stationary environment which can be in one of its states at each time step. The environment contains *r* devices that are hidden from the outside world. During each time step, each device *i* can be active or inactive with a fixed probability d_i independent from other devices. There is an agent interacting with the environment. At each time step, the agent selects *w* devices from the environment to use for its own purposes. However, the agent can only employ the active devices. After using the chosen devices, the agent releases its selected devices to the environment, and the environment activates or deactivates some of its devices for the next period. This procedure continues for an infinite number of iterations. The objective of the agent is to utilize as many devices as possible to fulfill its assigned tasks in time. Accordingly, it needs to find the *w* devices with the highest activation probabilities. Table 2 gives some instances of the introduced problem, which are used in the experiments of this section.

Table 2. 10 instances of the decision making problem. *r* and *w* respectively represent the number of available devices and the number of devices to be selected by the agent in each step. d_i is the activation probability of device *i*.

Prob.	**r,w**	**d_1**	**d_2**	**d_3**	**d_4**	**d_5**	**d_6**	**d_7**	**d_8**	**d_9**	**d_{10}**
1	6,3	0.7296	0.2987	0.3741	0.5187	0.1398	0.7555				
2	6,3	0.4780	0.1300	0.5339	0.5144	0.1665	0.6029				
3	7,3	0.7150	0.7125	0.9987	0.9173	0.3971	0.9228	0.1271			
4	7,3	0.9093	0.5233	0.2607	0.7955	0.9818	0.2673	0.1049			
5	8,3	0.8126	0.5963	0.0153	0.0855	0.0245	0.1393	0.2593	0.7124		
6	8,3	0.8095	0.3593	0.8131	0.5743	0.9497	0.5437	0.3369	0.0266		
7	9,5	0.5745	0.8156	0.3232	0.9225	0.8017	0.8098	0.0074	0.0617	0.6051	
8	9,5	0.0671	0.3588	0.6395	0.0236	0.0899	0.0505	0.5532	0.7165	0.2128	
9	10,4	0.6954	0.1900	0.3978	0.1777	0.5689	0.6425	0.1005	0.7337	0.6832	0.2827
10	10,4	0.6414	0.3879	0.1581	0.7081	0.1114	0.6702	0.3283	0.6734	0.8162	0.4271

In order to solve this problem with a multi-reinforcement LA, we consider an LA with *r* actions. At each time step, the LA selects *w* actions according to its internal state. Each selected action receives a reward if the corresponding device is active.

6.1.1. The sensitivity of the proposed models to the learning rate

In this section, the sensitivity of the proposed models to variations of their learning rates is investigated. In this regard, each model is examined with various settings for its learning rate on several instances of the described problem. The expected error of the learning agent is used as a performance metric in this section. First, we define the expected reward of each super-action $\underline{a_i}$ as follows:

$$\begin{aligned} &R_i = \sum\nolimits_{j \in I} d_j \\ &\text{with} \qquad\qquad\qquad\qquad (58) \\ &I = \left\{ j \mid a_j \in \underline{a_i} \right\}. \end{aligned}$$

If we represent the super-action set of the LA with $\underline{A}$, we can obtain the expected error of the LA at iteration *k* as follows:

$$\begin{aligned} &\chi(k) = \max\nolimits_{i \in I} \left(R_i\right) - \sum\nolimits_{i \in I} q_i(k) R_i \\ &\text{with} \qquad\qquad\qquad\qquad\qquad\qquad (59) \\ &I = \left\{1, \ldots, |\underline{A}|\right\}, \end{aligned}$$

where $q_i(k)$ is the selection probability of super-action $\underline{a_i}$ at step *k*.

Experiments 1: multi-reinforcement LA type III

Fig. (13) shows the obtained average results for multi-reinforcement LA III over 20 independent runs. Two major parameters affect the difficulty of each learning problem: the dimensionality of the problem and the closeness of the reward probabilities of different simple-actions. The difference between the third and the fourth highest reward probabilities is considerable in problem 5 (Table 2). As the LA needs to select three simple-actions in this problem instance, it can easily converge to the optimal super-action in a few iterations. In contrast, the difference between the third and the fourth reward probabilities is very small in problem 3. Accordingly in spite of its smaller action set, problem 3 requires more learning steps in comparison to problem 5.

As the learning rate increases from 0.001 to 0.1, the convergence speed of the algorithm increases significantly. However, the algorithm can examine fewer actions with a high learning rate, which may result in a premature convergence. Consequently, hard problems or problems with larger action sets require lower learning rates. For instance, using a high learning rate such as 0.1 resulted in the premature convergence of the algorithm in some of its executed runs on difficult problems 3 and 9. Based on the obtained results, a learning rate between 0.05 and 0.1 seems to be appropriate for most of the tested instances as the algorithm can find optimal or near optimal results in the designated number of iterations.

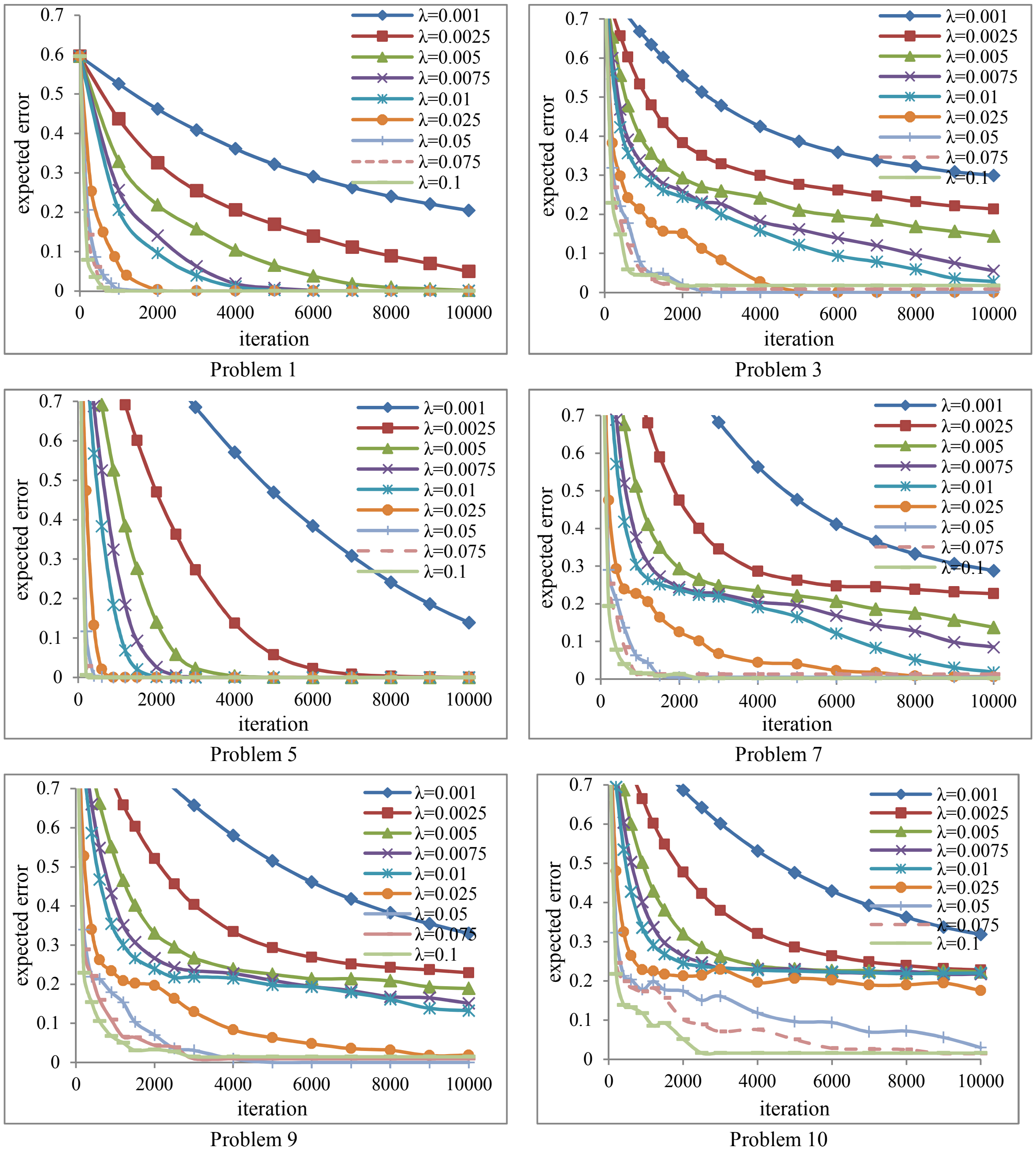


Figure 13. The average expected error curves of LA III on different problems from Table 2. The average expected errors of the algorithm on each problem instance are obtained using different settings of the learning rate (λ).

Experiment 2: multi-reinforcement LA type I

The same experiments as in the previous section are conducted using LA type I. As it is obvious from the achieved results in Fig. (14), LA type I demonstrates rather high convergence rates in comparison to LA type III on all of the tested problems, and besides problems 2 and 7 it is able to converge to the optimal super-action with all of the tested learning rates. Accordingly, LA I seems to be a better choice for random stationary environments in comparison to LA III; however, as it will be discussed in the subsequent sections LA III is able to obtain better results in multi agent settings.

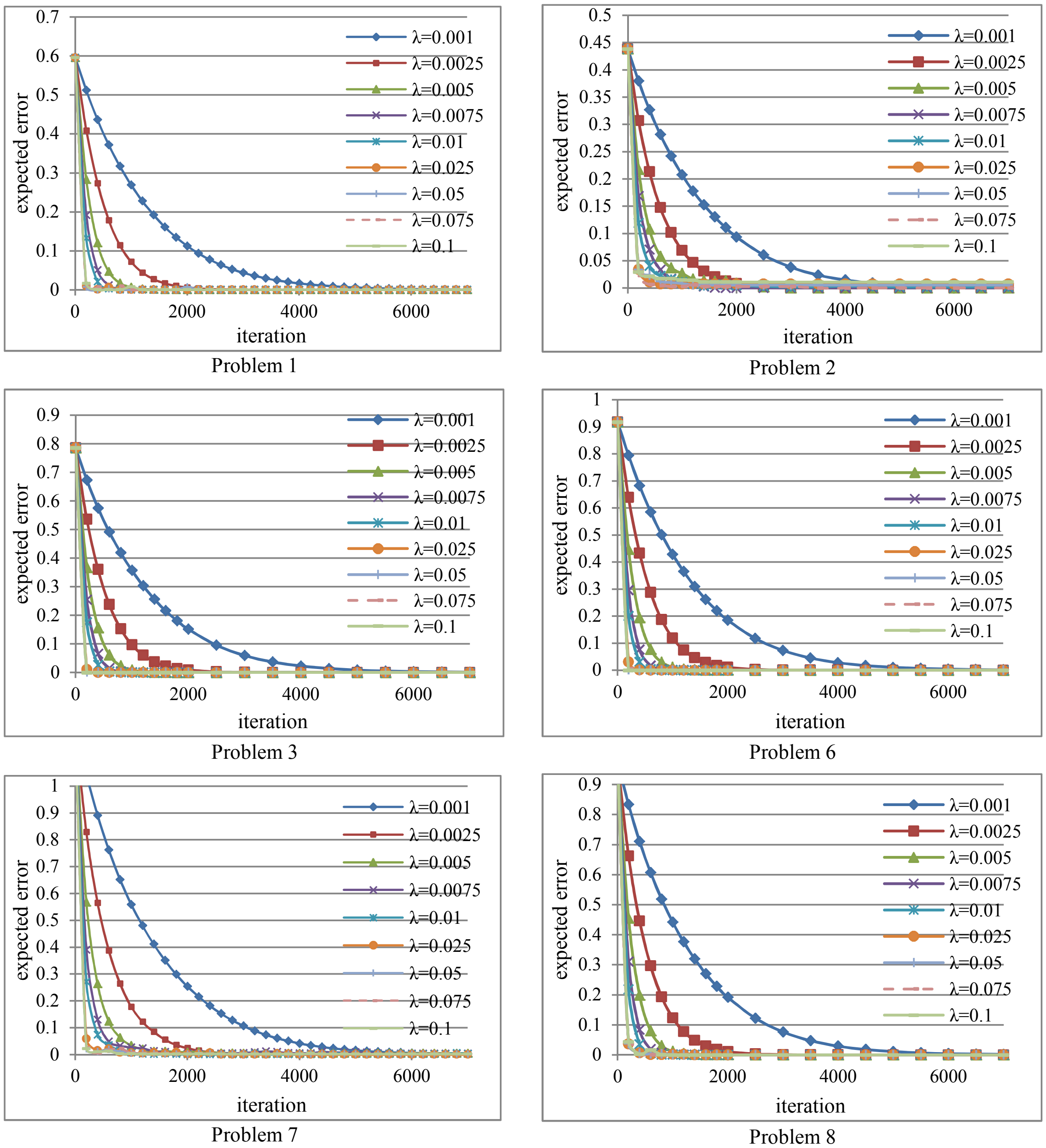
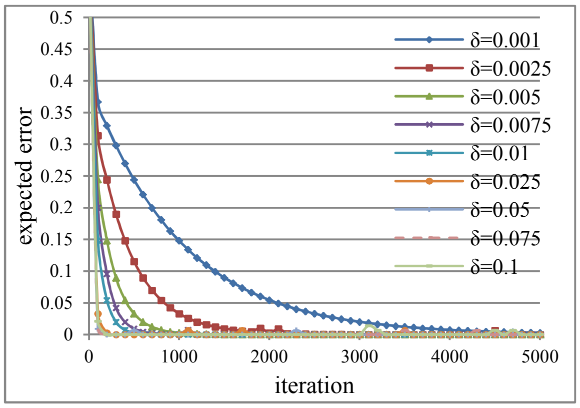


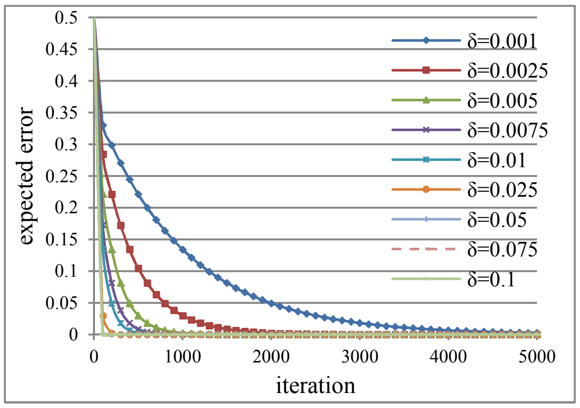
Figure 14. The average expected error curves of LA I on different problems from Table 2. The average expected errors of the algorithm on each problem instance are obtained using different settings of the learning rate (λ).

Experiment 3: multi-reinforcement LA type II

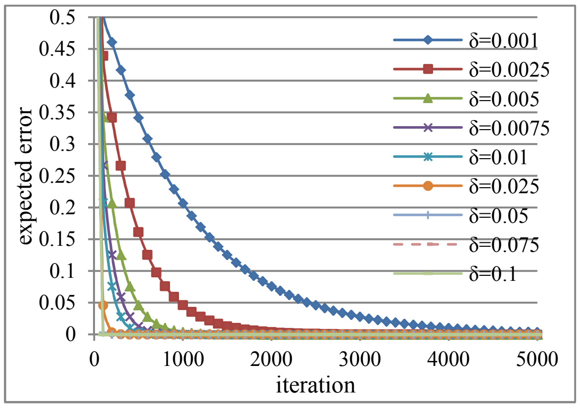
This section illustrates the behavior of the proposed multi-reinforcement LA II. In the experiments of this section, $\eta(k)$ is kept constant at 0.01, and an increasing sequence with an initial value of 0.1 is used as $(\lambda(k))_{k\in\mathbb{N}}$. Herein, we use a simple equation for defining the values of this parameter as follows $\lambda(k) = \lambda(k\text{-}1) + \delta(1\text{-} \lambda(k\text{-}1)$. The average expected errors of LA II on different problems with different settings for δ are provided in Fig. (15). The obtained results clearly demonstrate the effectiveness of the proposed learning scheme in finding the optimal subset of actions for the introduced problems. It can be observed that LA II is capable of finding the optimal super-actions on different problems in less than 1000 iterations with the settings of δ in the range [0.005,0.1]. Additionally, the algorithm can achieve expected errors very close to zero in a short period of time when $\delta \in$[0.025,0.1].



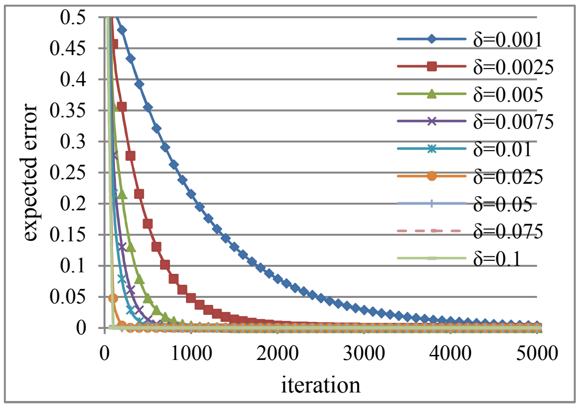
Problem 1



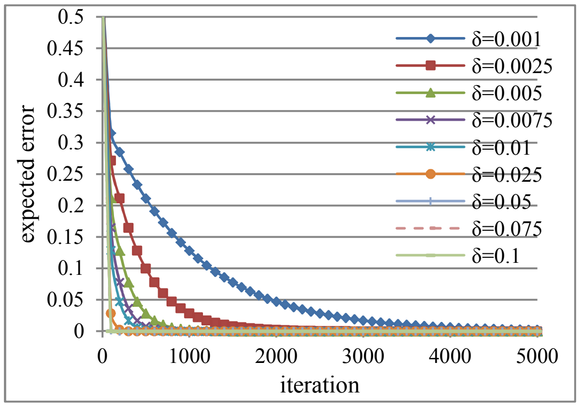
Problem 3



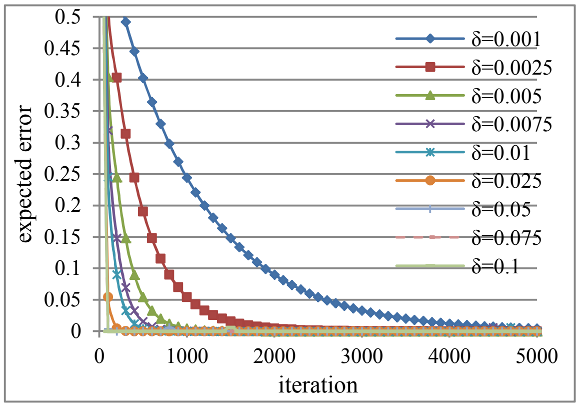
Problem 4



Problem 5



Problem 6



Problem 8

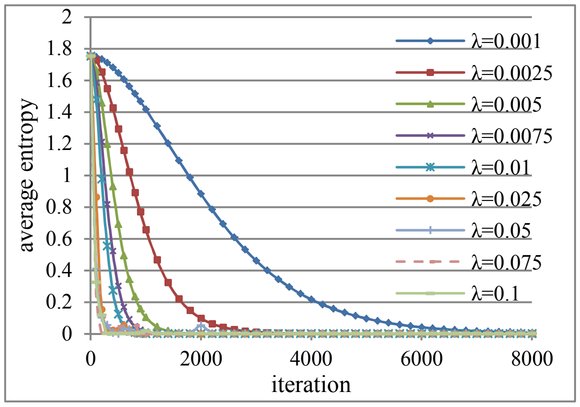
Figure 15. The average expected error curves of LA I on different problems from Table 2. The average expected errors of the algorithm on each problem instance are obtained using different settings of δ.

6.1.2. Convergence analysis

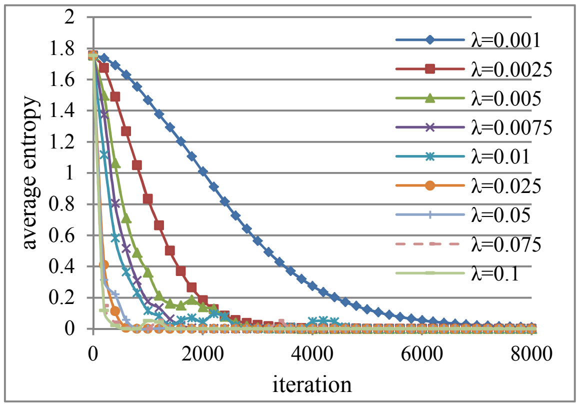
In order to investigate the convergence behavior of the proposed LA models, we use the concept of entropy, which was introduced in the context of information theory by Shannon [52]. If we consider the selected super-action of an LA during step k as a random variable, we can define its entropy using the following equation:

$$H(k)=\sum_{i=1}^{\binom{r}{w}} q(i,k).\ln\left(q(i,k)\right), \quad (60)$$

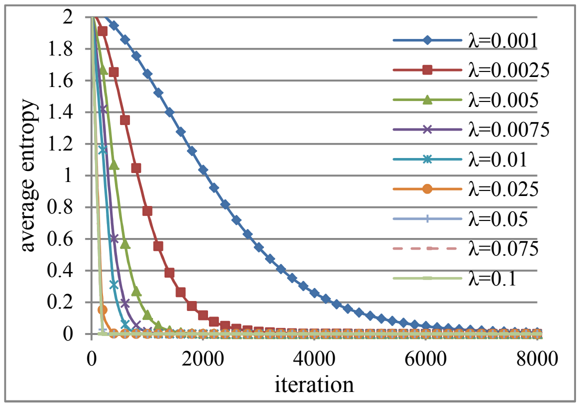
where $q(i,k)$ is the selection probability of the k^{th} super-action at time step k. $H(k)$ can be considered as a measure of uncertainty associated with the LA at step k; larger values of $H(k)$ represent more uncertainty in the decisions of the LA. The experiments of this section are conducted using LA I on different problems with different settings for the learning rate. Fig. (16) represents the average Entropy of the LA over 20 independent runs.



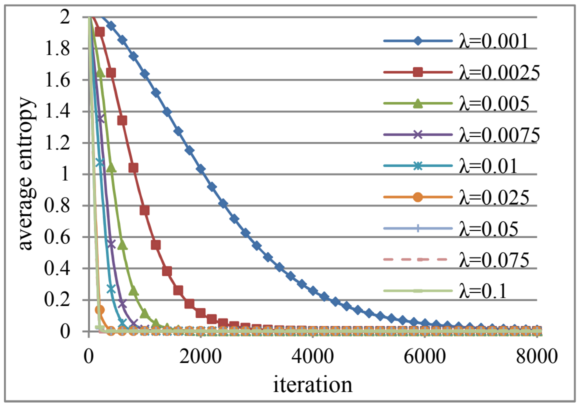
Problem 1



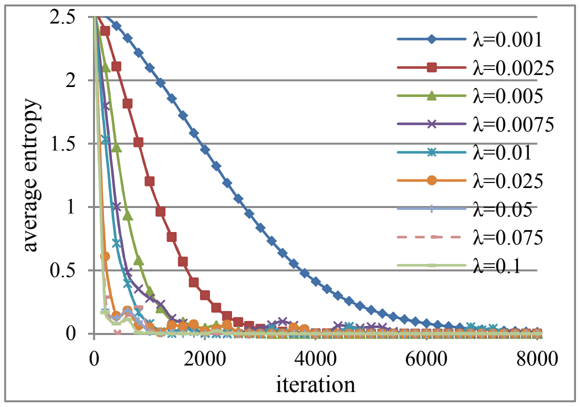
Problem 2



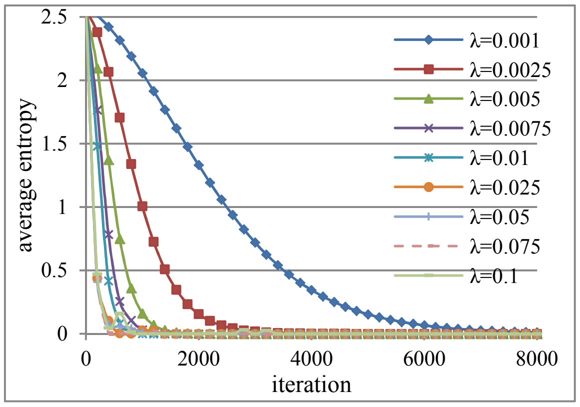
Problem 3



Problem 4



Problem 7



Problem 8

Figure 16. The average entropy of LA-I under different settings for the learning rate (λ) during 1E+4 iterations. The results are obtained over 20 independent runs.

Considering Fig. (16), it can be observed that the entropy is high at the beginning of the algorithm, but gradually decreases during the learning procedure. In all experiments, the average entropy converges to 0, meaning that the LA ultimately selects a particular super-action in every step. As demonstrated in the previous section, the expected error of the algorithm converges to 0 as well. Accordingly, it can be concluded that the proposed multi-reinforcement LA scheme always learns to select the optimal super-action with probability one.

6.1.3. Comparison of different introduced LA models

In this section, we compare the average final errors of the three proposed methods. Table 3 provides the average expected errors of the introduced LA models on the given problems after 1E+4 learning steps. The learning rate of all models is set to 0.075. Wilcoxon's rank sum test at a 5% significance level is also conducted to compare the algorithms [53]. The results corresponding to the conducted Wilcoxon's rank sum tests between LA I and LA II are given in the columns associated with LA II , where '+' represents that LA I is statistically better than LA II on the tested problem instance. When there is no significant difference between the two approaches '=' is utilized, and '-' denotes that LA II is statistically better than LA I. Similarly, the Wilcoxon's rank sum test results in the columns associated with LA III, compare LA I and LA II with LA III. It should be noted that the results smaller than 1E-8 are considered zero in the Table.

Considering the obtained results, the performances of different proposed approaches are very close. On all of the tested problems, the median of the final achieved results is zero, which means that the presented LA models were successful in finding the optimal super-action on most of their executed runs. However, on some problems like P3, P7, P9, and P10, some of the LA models were unsuccessful on some of their executed runs. We can resolve this issue by slightly decreasing the learning rate of the algorithms. Lowering the learning rate would prolong the convergence time of the algorithms, but can guarantee the acquisition of optimal solutions.

Table 3. Comparison results of different proposed LA schemes on the problems set of Table 2. Wilcoxon rank sum = WRS.

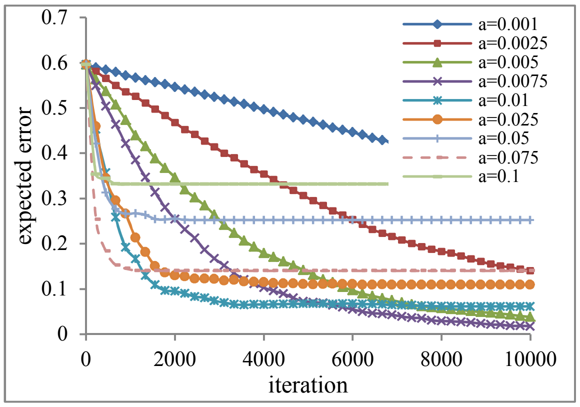
Prob.		**LA-I**	**LA-II**	**LA-III**	**Prob.**		**LA-I**	**LA-II**	**LA-III**
P1	mean	0.00E+00	0.00E+00	0.00E+00	**P6**	mean	0.00E+00	0.00E+00	0.00E+00
	median	0.00E+00	0.00E+00	0.00E+00		median	0.00E+00	0.00E+00	0.00E+00
	std	0.00E+00	0.00E+00	0.00E+00		std	0.00E+00	0.00E+00	0.00E+00
	WRS		(=)	(=,=)		WRS		(=)	(=,=)
P2	mean	0.00E+00	1.19E-02	1.73E-03	**P7**	mean	1.53E-03	1.53E-03	1.07E-02
	median	0.00E+00	0.00E+00	0.00E+00		median	0.00E+00	0.00E+00	0.00E+00
	std	0.00E+00	1.91E-02	7.94E-03		std	6.84E-03	6.84E-03	1.50E-02
	WRS		(+)	(=,-)		WRS		(=)	(+,+)
P3	mean	0.00E+00	0.00E+00	0.00E+00	**P8**	mean	0.00E+00	0.00E+00	0.00E+00
	median	0.00E+00	0.00E+00	0.00E+00		median	0.00E+00	0.00E+00	0.00E+00
	std	0.00E+00	0.00E+00	0.00E+00		std	0.00E+00	0.00E+00	0.00E+00
	WRS		(=)	(=,=)		WRS		(=)	(=,=)
P4	mean	0.00E+00	0.00E+00	0.00E+00	**P9**	mean	0.00E+00	0.00E+00	3.68E-03
	median	0.00E+00	0.00E+00	0.00E+00		median	0.00E+00	0.00E+00	0.00E+00
	std	0.00E+00	0.00E+00	0.00E+00		std	0.00E+00	0.00E+00	1.65E-02
	WRS		(=)	(=,=)		WRS		(=)	(+,+)
P5	mean	0.00E+00	0.00E+00	0.00E+00	**P10**	mean	0.00E+00	0.00E+00	1.23E-02
	median	0.00E+00	0.00E+00	0.00E+00		median	0.00E+00	0.00E+00	0.00E+00

	std WRS	0.00E+00	0.00E+00 (=)	0.00E+00 (=,=)		std WRS	0.00E+00	0.00E+00 (=)	1.55E-02 (+,+)

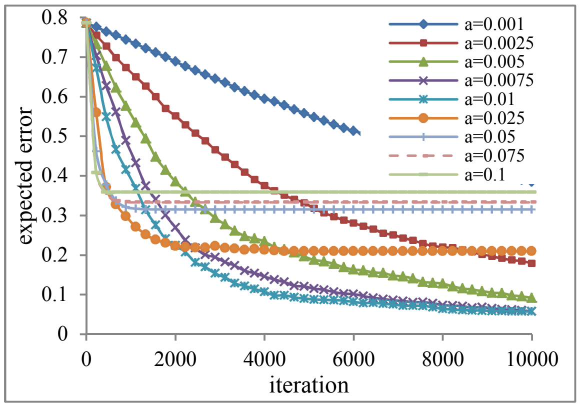
6.1.4. Comparison between LA III and L_{RI}

In order to illustrate how using multiple reinforcements can improve the performance of a learning approach, this section compares LA III with L_{RI} (see section 2.1) on the introduced decision making problems. Fig. (17) provides the average expected errors for the L_{RI} algorithm under different settings of its learning rate. As it can be seen from the achieved results, L_{RI} achieves its best results in the given time when it is using a learning rate near 0.0075. The algorithm suffers from premature convergence on the tested problems when its learning rate is set to a value higher than 0.0075. Moreover, it is occasionally unable to approach the optimal solutions with appropriate precisions within 10000 time steps. This issue is more apparent when the learning rate of the algorithm is lower than 0.0025.

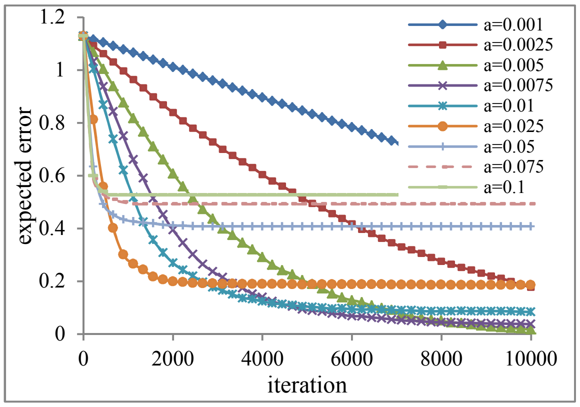
Fig. (18) compares LA III with L_{RI} on a set of problem instances from Table 2. We compared the algorithms under their best settings for their learning rates. It can be observed from the achieved results that LA III demonstrates superior convergence rates in comparison to L_{RI}. The obtained expected errors of the proposed approach in 2000 iterations are much lower than those that are obtained by the L_{RI} algorithm after 10000 iterations. On problems such as P1, LA III even obtains superior results in comparison to L_{RI} in less than 1000 iterations. LA III also demonstrates a superior performance in terms of the accuracy of its obtained final solutions. Its achieved expected errors are almost zero on all of the tested instances, while the obtained expected errors of L_{RI} on problems such as P9 are considerable.



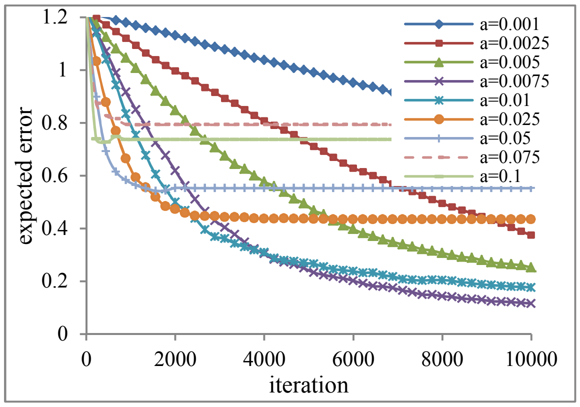
Problem 1



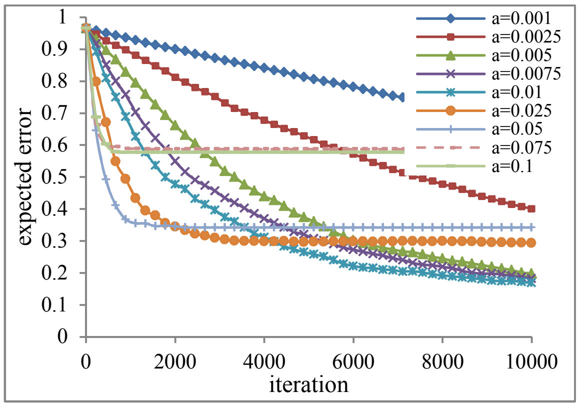
Problem 3



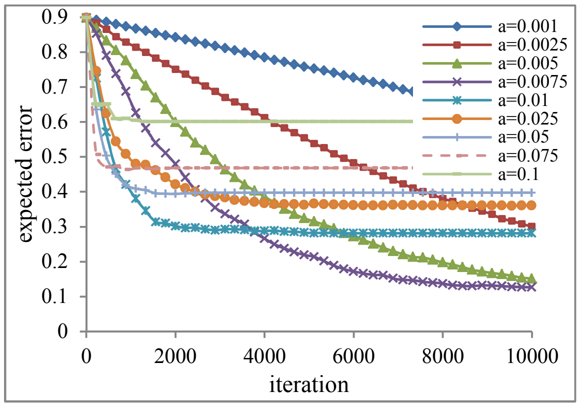
Problem 5



Problem 7

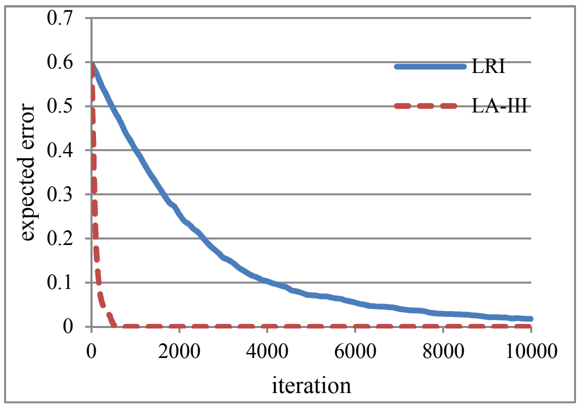


Problem 9

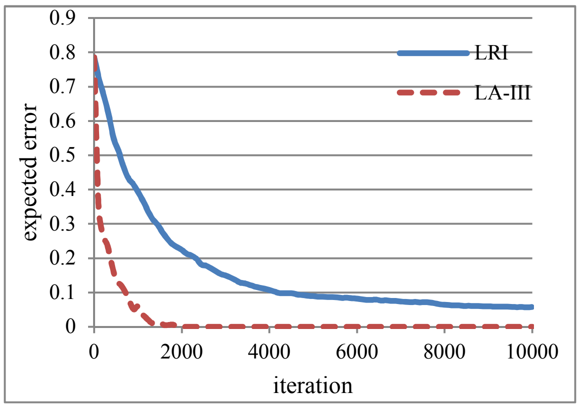


Problem 10

Figure 17. The average expected error curves of L_{RI} on different problems from Table 2. The average expected errors of the algorithm on each problem instance are obtained using different settings of the learning rate (a).



Problem 1



Problem 3

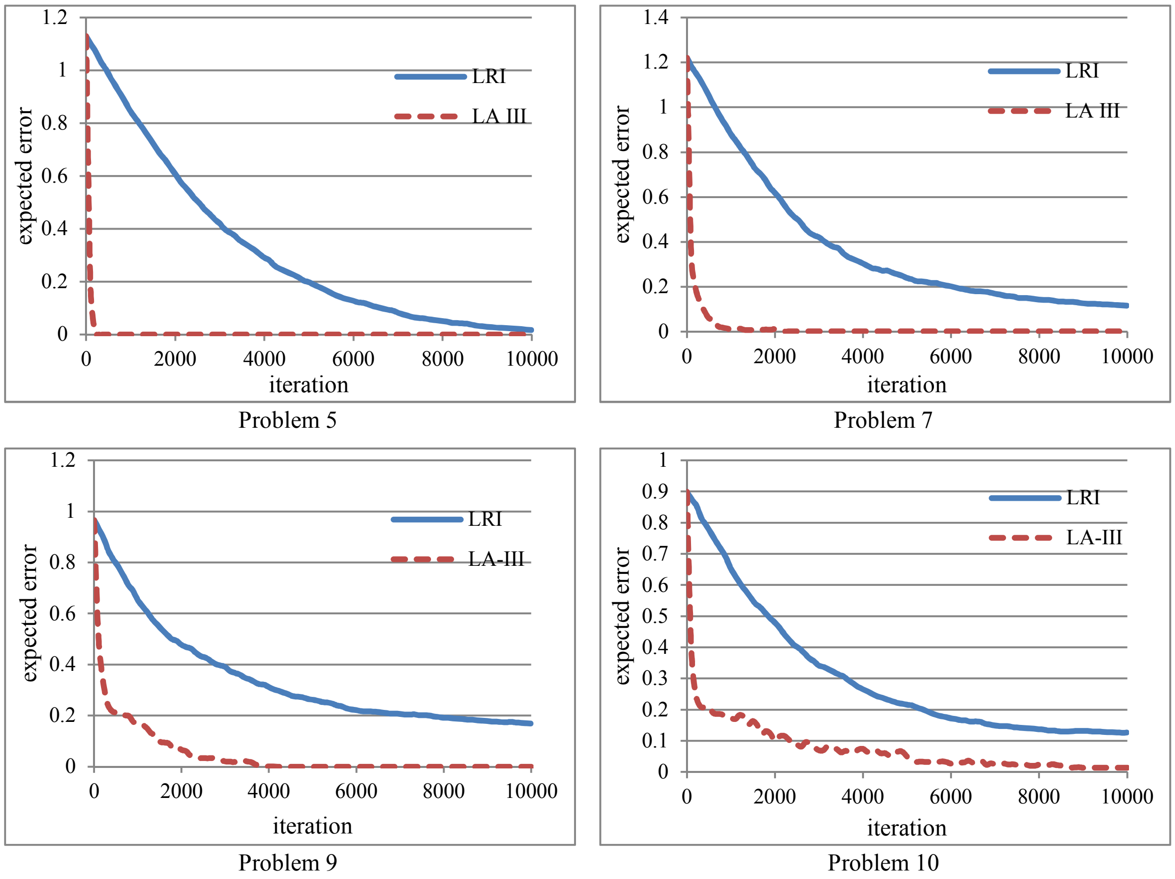


Figure 18. The average expected error curves of LA-III and L_{RI} during 1E+4 iterations on a set of problems from Table 2.

6.2. Case study: channel assignment in multi radio wireless networks using the proposed CLA models

In this section we examine the performance of the CLA models on the channel assignment problem described in section 5. Similar to [54-55], we consider random networks for performance evaluations. In this regard, two networks are generated by randomly placing 20 and 25 nodes in a 1000×1000 square meters of area. The associated conflict graphs of these networks are provided in Fig. (10) and Fig. (19). Moreover, a bidirectional single-hop flow between each pair of users is considered which has a constant bit rate.

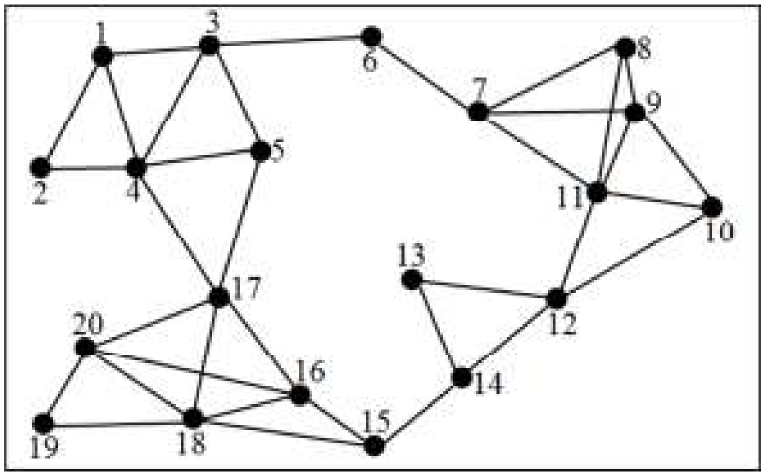


Figure 19. A sample conflict graph of a wireless network with 20 pairs of users

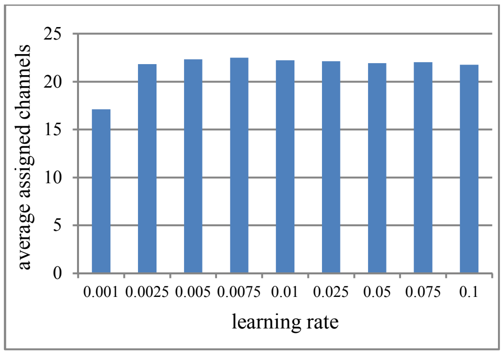
In order to evaluate the performance of the algorithms, we use the total number of none conflicting channels that are assigned to the different nodes in a network. Using this number, one can also obtain the total throughput of the network. Five instances of each network are considered in this section. These instances are summarized in Table 4. For the rest of the paper, the network depicted in Fig. (19) will be denoted by PA, and its instances will be identified by PA-instance_number. Similarly, the network in Fig. (10) and its instances will be denoted by PB and PB-instance-number, respectively.

Table 4. Different instances of the channel assignment problems. Each instance determines the number of radios and the number of channels for each node in the considered networks.

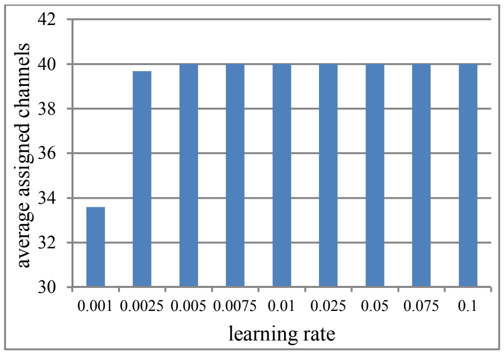
	Instance 1	**Instance 2**	**Instance 3**	**Instance 4**	**Instance 5**
Channel Num	3	12	12	12	12
Radio Num	2	2	4	6	8

6.2.1. The sensitivity of the proposed CLA models to the learning rate

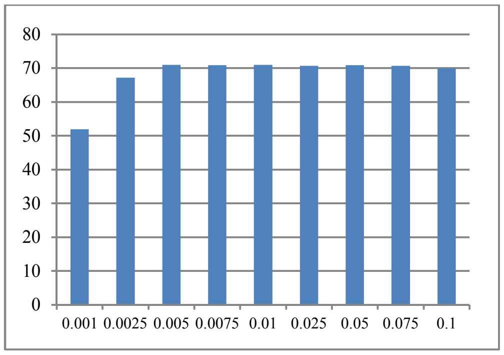
This section investigates the effects of the learning rate on the CLA model which is based on multi-reinforcement LA III. As the results of the other developed models are much similar, they are excluded for the sake of space economy. The results of this section are obtained over 25 independent runs on the different instances of PA, and each run of the algorithm on a particular instance is terminated after 1E+4 iterations. Considering Fig. (20), the proposed model obtains inferior results when its learning rate is 0.001. Using this low learning rate prolongs the convergence time of the algorithm since the action probabilities are modified in very small steps. Additionally, the proposed model demonstrates a rather weaker performance under $\lambda=0.1$ on some instances like PA-5, which is due to premature convergence without sufficient exploration of actions. Besides the settings with $\lambda \in \{0.001, 0.025, 0.1\}$, the algorithm obtains very close results under the other experimented settings. For the rest of the paper, a learning rate of 0.01 will be used for all of the proposed CLA models as it exhibits appropriate convergence behavior. Moreover, using a small learning rate (λ=0.01) enables the algorithm to examine different actions for a sufficient number of times before convergence.



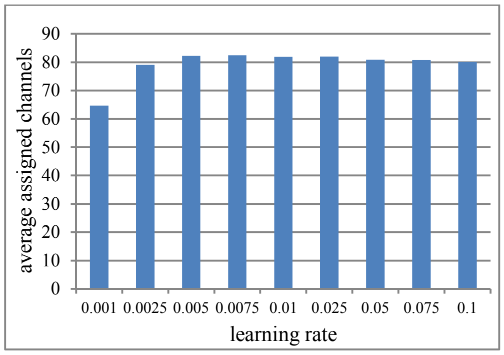
Problem instance 1



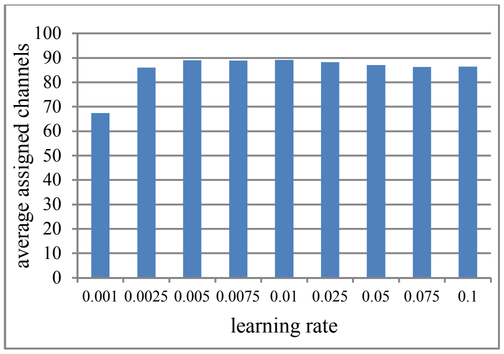
Problem instance 2



Problem instance 3



Problem instance 4



Problem instance 5

Figure 20. The sensitivity of the CLA with multi-reinforcement LA III under different settings of learning rate on PA.

6.2.2. Comparison results of different CLA models

The results of the different CLA methods on the channel assignment problems are given in Table 5. The CLA model introduced in section 4 is denoted by M-CLA. The statistical results of each method on each one of the considered problem instances is obtained over 25 independent runs. As it is obvious from Table 5, all CLA models have very close results. However, the CLA models based on multi-reinforcement LA III and maximum expected reward exhibit fairly better performances. LA III is mainly based on L_{RI} learning model. Although the theoretical characteristics of CLA-LA-III are not considered in this paper, the convergence properties of CLA with L_{RI} learning scheme are investigated in several works, where some appropriate convergence behaviors are proven for this model [26].

Problem instance 1 uses 3 orthogonal channels with two radios for each user. Accordingly, at most 40 and 50 non-conflicting assignments are possible for problems PA and PB, respectively. However, as it can be observed from the related conflict graphs of the problems, many nodes have neighborhoods larger than 2. Accordingly, non-conflicting assignments may be impossible for some radios. Scenario 2 is a very easy problem, and all methods can achieve an optimal channel assignment as there are 12 different channels with only 2 radios in each node. This problem instance gets harder in scenarios 3, 4 and 5 as we increase the number of radios of each user.

Although in terms of convergence rate, a single LA III demonstrates rather weaker performances in comparison to the other developed LAs (see section 6.1.1), a group of LA III learning agents exhibits better performances in a CLA. CLA-LA-III has obtained slightly better results on 8 problem instances in comparison to CLA-LA-I. It also obtained better results on 7 problems in comparison to CLA-LA-II. Accordingly, LA III seems to be a preferable choice for the CLA model. Moreover, CLA-LA-II is slightly better than CLA-LA-I, which is mainly due to the properties that are discussed in section 3.2. M-CLA obtains the best results on each problem instance; however, its obtained results are very close to the ones that are achieved by CLA-LA-III. It should be noted that CLA-LA-III is a very simple and straight-forward model, which can be easily implemented in a distributed environment.

Table 5. Comparison results of the different proposed CLA models on the Channels assignment problems in terms of the mean, median, and std of the obtained solutions.

P	**CLA-LA-I**	**CLA-LA-II**	**CLA-LA-III**	**M-CLA**
PA-1	2.13E+01	2.22E+01	2.22E+01	2.28E+01
	2.10E+01	2.20E+01	2.20E+01	2.30E+01
	8.91E-01	7.07E-01	5.97E-01	4.36E-01
PA-2	4.00E+01	4.00E+01	4.00E+01	4.00E+01
	4.00E+01	4.00E+01	4.00E+01	4.00E+01
	0.00E+00	0.00E+00	0.00E+00	0.00E+00
PA-3	6.98E+01	7.04E+01	7.12E+01	7.13E+01
	7.00E+01	7.00E+01	7.10E+01	7.10E+01

	6.24E-01	6.45E-01	6.88E-01	4.76E-01
PA-4	8.02E+01	8.12E+01	8.21E+01	8.24E+01
	8.00E+01	8.10E+01	8.20E+01	8.20E+01
	1.11E+00	7.64E-01	7.81E-01	5.77E-01
PA-5	8.60E+01	8.79E+01	8.96E+01	8.97E+01
	8.60E+01	8.80E+01	8.90E+01	9.00E+01
	1.14E+00	1.22E+00	7.68E-01	6.90E-01
PB-1	2.91E+01	2.92E+01	2.98E+01	3.00E+01
	2.90E+01	2.90E+01	3.00E+01	3.00E+01
	5.72E-01	6.88E-01	4.08E-01	0.00E+00
PB-2	5.00E+01	5.00E+01	5.00E+01	5.00E+01
	5.00E+01	5.00E+01	5.00E+01	5.00E+01
	0.00E+00	0.00E+00	0.00E+00	0.00E+00
PB-3	9.46E+01	9.53E+01	9.70E+01	9.71E+01
	9.50E+01	9.50E+01	9.70E+01	9.70E+01
	1.08E+00	1.07E+00	7.90E-01	7.26E-01
PB-4	1.10E+02	1.12E+02	1.13E+02	1.13E+02
	1.10E+02	1.13E+02	1.13E+02	1.13E+02
	1.21E+00	8.79E-01	9.80E-01	4.58E-01
PB-5	1.16E+02	1.17E+02	1.18E+02	1.19E+02
	1.16E+02	1.17E+02	1.18E+02	1.18E+02
	1.77E+00	1.01E+00	1.16E+00	8.43E-01

6. 2. 3. Comparison with other approaches

In this section, we compare the proposed multi-reinforcement CLA models on the described channel assignment problems with two approaches introduced in [55-56]. The total payoff, as described in [50], will be employed as a performance metric in comparisons. This criterion is also closely related to the throughput of the system. Let Y_{ic} denote the number of radios in the neighborhood of node I which are tuned on channel c; it should be noted that node i is also counted in Y_{ic} if it has a radio on channel c. The maximum data rate which can be achieved in the collision domain of node i on channel c can be denoted by $\tau_{max}(Y_{ic})$. In practice, $\tau_{max}(Y_{ic})$ is a non-increasing function of Y_{ic}. However, for the rest of the paper, $\tau_{max}(Y_{ic})=\tau$ will be considered independent of Y_{ic} and c. Additionally, it will be assumed that $\tau_{max}(Y_{ic})$ is shared equally among the interfering radios which are assigned to channel c [56]. Based on these assumptions, the utility or payoff of a node like i can be obtained according to the following equation [56]:

$$u_i = \sum\nolimits_{c \in S_i} \tau_{ic}$$

with: (61)

$$\tau_{ic} = \begin{cases} \dfrac{1}{Y_{ic}}\tau & \text{if node } i \text{ has a radio on channel } c \\ 0 & \text{otherwise} \end{cases}$$

According to the defined utility for the nodes, the total payoff or the total utility of the network can be defined as:

$$U = \sum_{i=1:N} u_i \qquad (62)$$

Herein, the compared methods corresponding to Algorithm 1 presented in [56] and algorithm 2 presented in [55] will be denoted by CAL1 and CAL2, respectively. Fig. (21) gives the comparison results based on the total payoff criteria. It should be noted that we only considered the non-conflicting assigned channels in the computation of the total utility in the proposed CLA models. In the results, τ is set to 54Mbps.

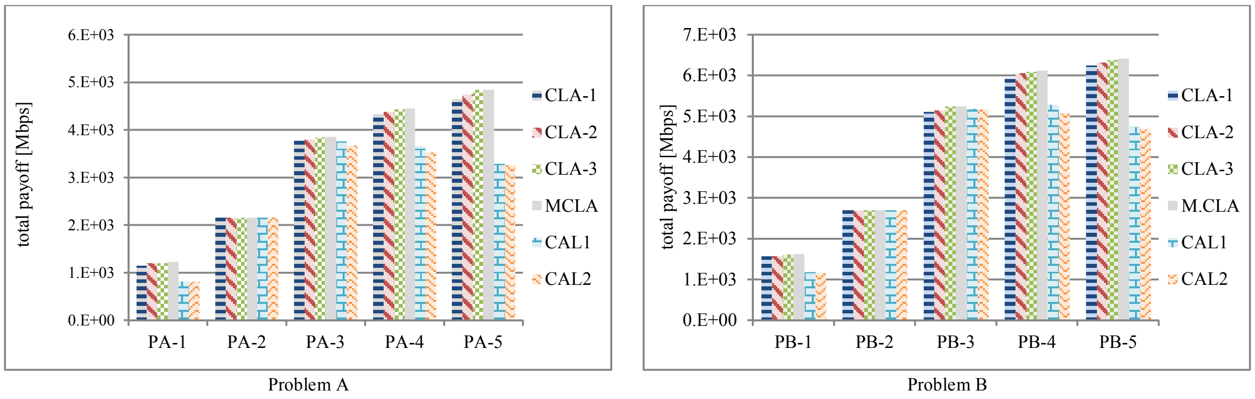


Figure 21. Total payoff of the compared methods on the different instances of the channel allocation problems.

As it can be noticed from the obtained results, the proposed CLA models can be effectively utilized for the given channel assignment problem. One advantage of the CLA models over the two other compared approaches is the synchronous nature of their updating procedures. In the CLA models at each instant of time, all nodes choose their channels and update their variables simultaneously. Accordingly, all nodes can be updated for several times in a short period which significantly reduces the convergence time of the algorithms. However, although the two compared methods are distributed, they update the network nodes asynchronously and mostly one node at each time step. Accordingly, for a network consisting of N nodes, CLA approaches update each node $o(N)$ times more in a same period of time when they are compared with CLA1 and CAL2. In addition, the comparisons based on the total payoff criterion, demonstrate the superiority of the CLA approaches over CLA1 and CLA2. CAL1 and CLA2 achieve identical or close solutions on the second and third instances of the two considered networks. However, they attain lesser average total payoffs on the other hard instances of the described problems. Considering PA-1, PA-4, PA-5, PB-1, BP-4, and PB-5, there are significant differences between the achieved results of the proposed CLA based methods and CAL1 and CAL2.

7. Discussion

Distributed decision making arises in many areas of computer science. In such systems, a decision making entity may have its own objectives and learns about the objectives of other entities through their feedback signals. The proposed multi-reinforcement schemes are suitable for such environments, where each entity interacts with several other entities via its limited number of actuators and receives multiple feedbacks for its performed actions (made decisions). The introduced LA models can simultaneously perform several actions and receive multiple feedbacks on their chosen actions. These feedbacks should reflect the favorability of each performed action so that the proposed learning algorithms can adapt their decision policies for future.

The developed CLA schemes are closed, meaning that each LA interacts only with other LAs in its neighborhood and adjusts its decision policies based on these interactions. However, there are many applications where the actions performed by the interconnected learning entities also affect a global environment or other systems in the environment. As a result, the global environment may change its current state according to the executed actions. Accordingly, the decision making policies of the interconnected entities should take the state of the global environment into consideration, which is not supported by the current developed models. Additionally, we assume that there is a one to one correspondence between the executed actions by an agent and its received reinforcements. However, other patterns for the feedbacks might be possible. For instance, a signal could be related to several actions to represent that these actions should not be selected at the same time.

This paper investigated the application of the proposed approaches on a channel assignment problem. Other areas of application for the introduced models could be sensor networks, internet of things, and cyber physical systems. For instance, CLA would be an appropriate model for cyber physical systems composed of a large number of interconnected and embedded components. A cyber physical system is an integration of computation with physical processes that provides new capabilities to the systems being able to interact with a physical world. The uncertainty in physical environments and the existence of multiple decision-making entities with different objectives make a cyber physical system to be more reliant on machine learning approaches. The objective function of each agent in these systems may be unknown by the others, and each agent would receive other agents' feedbacks for training purposes. The accumulation of multiple feedbacks from different sources was one of the motivations of the proposed multi-reinforcement learning schemes.

8. Conclusion

This paper investigates new LA and CLA models which are based on multiple reinforcement signals. Each LA in the proposed models aims at learning the optimal subset of its actions using multiple feedbacks from its environment. Using a separate reinforcement for each chosen action enables the LA to learn the effectiveness of its different actions in the chosen subsets in parallel. This kind of decision making is useful in scenarios like multi-agent environments where each agent has limited resources and actuators and performs several tasks simultaneously via its actuators. The convergence behavior of some of the proposed models is theoretically analyzed. It is shown that the proposed CLA model named MCLA, which is based on the idea of maximizing the expected rewards for each LA, converges to a compatible point when it uses potential local rules. Also, it is discussed that two of the proposed LA models are ε-optimal; however, neither of these two models is absolutely expedient. The absolute expediency of the third proposed LA model in the paper is not proved, and accordingly cannot be guaranteed. The absolute expediency is one of the main characteristics that can be very profitable for LA algorithms operating in a CLA structure in multi agent environments. However, the three proposed LA models proved to be effective in CLA when they are tested on a decentralized channel assignment problem. In contrast to many channel assignment approaches that are central, the introduced methods in this paper are completely distributed and each node governs its own channel assignment without any cooperation. Moreover, the experimental studies demonstrate that the three developed LA models can achieve very close performances to MCLA when they are integrated in a CLA structure. As the developed LA models are very simple and have very low computational costs, they can be easily employed in the decentralized and distributed problems.

Although the works in this paper are suitable for real world applications where the agents need to perform several decisions at each time instant; however, there are some shortcomings needed to be addressed in the future works. First, the design and the proof of an absolutely expedient multi-reinforcement LA is necessary when we are dealing with the CLA approach. Next, hybridized learning methods should be considered for dealing with situations where some feedback signals from the environment affect several actions in the chosen action subsets. These environmental feedbacks would represent the correlations between several performed actions.

References

[1] K.S. Narendra, M.A. Thathachar, Learning Automata: an Introduction, Courier Corporation, 2012.
[2] M.A. Thathachar, P.S. Sastry, Networks of learning automata: Techniques for online stochastic optimization, Springer Science & Business Media, 2011.
[3] A. Moayedikia, K.-L. Ong, Y.L. Boo, W.G. Yeoh, Task assignment in microtask crowdsourcing platforms using learning automata, Eng. Appl. Artif. Intell. 74 (2018) 212-225.
[4] B. Moradabadi, M.R. Meybodi, Link prediction in weighted social networks using learning automata, Eng. Appl. Artif. Intell. 70 (2018) 16-24.
[5] G. Lingam, R.R. Rout, D.V. Somayajulu, Learning automata-based trust model for user recommendations in online social networks, Comput. Electr. Eng. 66 (2018) 174-188.

[6] J. Zhu, W. Gu, G. Lou, L. Wang, B. Xu, M. Wu, W. Sheng, Learning automata-based methodology for optimal allocation of renewable distributed generation considering network reconfiguration, IEEE Access, 5 (2017) 14275-14288.
[7] J. Zhang, C. Wang, M. Zhou, Fast and epsilon-optimal discretized pursuit learning automata, IEEE Trans. Cybern. 45 (2015) 2089-2099.
[8] J. Zhang, C. Wang, M. Zhou, Last-position elimination-based learning automata, IEEE Trans. Cybernetics, 44 (2014) 2484-2492.
[9] A. Yazidi, B.J. Oommen, G. Horn, O.-C. Granmo, Stochastic discretized learning-based weak estimation: a novel estimation method for non-stationary environments, Pattern Recognit. 60 (2016) 430-443.
[10] C. Szepesvári, M.L. Littman, A unified analysis of value-function-based reinforcement-learning algorithms, Neural Comput. 11 (1999) 2017-2060.
[11] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, Human-level control through deep reinforcement learning, Nature, 518 (2015) 529-533.
[12] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971, (2015).
[13] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, B. Coppin, Deep reinforcement learning in large discrete action spaces, arXiv preprint arXiv:1512.07679, (2015).
[14] H. Beigy, M.R. Meybodi, Asynchronous cellular learning automata, Automatica, 44 (2008) 1350-1357.
[15] S. Yun, J. Choi, Y. Yoo, K. Yun, J.Y. Choi, Action-Driven Visual Object Tracking With Deep Reinforcement Learning, IEEE Trans. Neural Networks Learn. Syst. 29 (2018) 2239-2252.
[16] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, P. de la Puente, P. Campoy, A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform, J. Intell. Rob. Syst. (2018), https://doi.org/10.1007/s10846-018-0891-8.
[17] A. Waldock, C. Greatwood, F. Salama, T. Richardson, Learning to Perform a Perched Landing on the Ground Using Deep Reinforcement Learning, J. Intell. Rob. Syst. (2017), https://doi.org/10.1007/s10846-017-0696-1.
[18] I. Carlucho, M. De Paula, S. Wang, Y. Petillot, G.G. Acosta, Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning, Rob. Auton. Syst. 107 (2018) 71-86.
[19] J. Qiao, G. Wang, W. Li, M. Chen, An adaptive deep Q-learning strategy for handwritten digit recognition, Neural Networks, 107 (2018) 61-71.
[20] M. Mahmud, M.S. Kaiser, A. Hussain, S. Vassanelli, Applications of deep learning and reinforcement learning to biological data, IEEE Trans. Neural Networks Learn. Syst. 29 (2018) 2063-2079.
[21] H. Li, R. Cai, N. Liu, X. Lin, Y. Wang, Deep reinforcement learning: Algorithm, applications, and ultra-low-power implementation, Nano Commun. Networks, 16 (2018) 81-90.
[22] H. Beigy, M.R. Meybodi, Open synchronous cellular learning automata, Adv. Complex Syst. 10 (2007) 527-556.
[23] H. Beigy, M.R. Meybodi, Cellular learning automata with multiple learning automata in each cell and its applications, IEEE Trans. Syst. Man Cybern. Part B Cybern. 40 (2010) 54-65.
[24] A.M. Saghiri, M.R. Meybodi, Open asynchronous dynamic cellular learning automata and its application to allocation hub location problem, Knowl. Based Syst. 139 (2018) 149-169.
[25] M. Esnaashari, M.R. Meybodi, Irregular cellular learning automata, IEEE Trans. Cybern. 45 (2015) 1622-1632.
[26] H. Beigy, M.R. Meybodi, A mathematical framework for cellular learning automata, Adv. Complex Syst. 7 (2004) 295-319.
[27] M. Ahangaran, N. Taghizadeh, H. Beigy, Associative cellular learning automata and its applications, Appl. Soft Comput. 53 (2017) 1-18.
[28] M. Esnaashari, M.R. Meybodi, Dynamic irregular cellular learning automata, J. Comput. Sci. 24 (2018) 358-370.
[29] B. Moradabadi, M.R. Meybodi, Wavefront cellular learning automata, Chaos Interdiscip. J. Nonlinear sci. 28 (2018) 021101.
[30] A.M. Saghiri, M.R. Meybodi, An approach for designing cognitive engines in cognitive peer-to-peer networks, J. Network Comput. Appl. 70 (2016) 17-40.
[31] A.M. Saghiri, M.R. Meybodi, A closed asynchronous dynamic model of cellular learning automata and its application to peer-to-peer networks, Genet. Program. Evolvable Mach. 18 (2017) 313-349.
[32] R. Vafashoar, M.R. Meybodi, Multi swarm bare bones particle swarm optimization with distribution adaption, Appl. Soft Comput. 47 (2016) 534-552.
[33] R. Vafashoar, M.R. Meybodi, Cellular learning automata based bare bones PSO with maximum likelihood rotated mutations, Swarm Evol. Comput. (2018), https://doi.org/10.1016/j.swevo.2018.08.016.
[34] Y. Zhao, W. Jiang, S. Li, Y. Ma, G. Su, X. Lin, A cellular learning automata based algorithm for detecting community structure in complex networks, Neurocomputing, 151 (2015) 1216-1226.
[35] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Finding the shortest path in stochastic graphs using learning automata and adaptive stochastic petri nets, Int. J. Uncertainty Fuzziness Knowledge Based Syst. 25 (2017) 427-455.
[36] M. Ghavipour, M.R. Meybodi, Irregular cellular learning automata-based algorithm for sampling social networks, Eng. Appl. Artif. Intell. 59 (2017) 244-259.
[37] M.M.D. Khomami, A. Rezvanian, M.R. Meybodi, A new cellular learning automata-based algorithm for community detection in complex social networks, J. Comput. Sci. 24 (2018) 413-426.
[38] M. Khani, A. Ahmadi, H. Hajary, Distributed task allocation in multi-agent environments using cellular learning automata, Soft Comput. (2017), https://doi.org/10.1007/s00500-017-2839-5.
[39] M. Esnaashari, M. Meybodi, A cellular learning automata based clustering algorithm for wireless sensor networks, Sens. Lett. 6 (2008) 723-735.
[40] F. Belletti, D. Haziza, G. Gomes, A.M. Bayen, Expert level control of ramp metering based on multi-task deep reinforcement learning, IEEE Trans. Intell. Transp. Syst. 19 (2018) 1198-1207.
[41] J. Zhang, Z. Li, Q. Kang, M. Zhou, A new class of learning automata for selecting an optimal subset, in: Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on, IEEE, 2014, pp. 3429-3434.
[42] X. Guo, W. Jiang, H. Ge, S. Li, A New Learning Automata Algorithm for Selection of Optimal Subset, in: International Conference on Intelligent Computing, Springer, 2015, pp. 693-702.
[43] H. Morshedlou, M.R. Meybodi, A new local rule for convergence of ICLA to a compatible point, IEEE Trans. Syst. Man Cybern. Syst. 47 (2017) 3233-3244.
[44] D. Monderer, L.S. Shapley, Potential games, Games Econ. Behav. 14 (1996) 124-143.
[45] S. Wolfram, A New Kind of Science, Wolfram media Champaign, 2002.
[46] Y. Babichenko, O. Tamuz, Graphical potential games, J. Econ. Theory, 163 (2016) 889-899.

[47] L.E. Ortiz, Graphical potential games, arXiv preprint arXiv:1505.01539, (2015).
[48] P.B. Duarte, Z.M. Fadlullah, A.V. Vasilakos, N. Kato, On the partially overlapped channel assignment on wireless mesh network backbone: A game theoretic approach, IEEE J. Sel. Areas Commun. 30 (2012) 119-127.
[49] D.S. Leslie, E.J. Collins, Generalised weakened fictitious play, Games Econ. Behav. 56 (2006) 285-298.
[50] M. Felegyhazi, M. Cagalj, S. Saeedi Bidokhti, J.-P. Hubaux, Non-cooperative multi-radio channel allocation in wireless networks, in: Infocom 2007, 2007.
[51] Z. Fang, B. Bensaou, Fair bandwidth sharing algorithms based on game theory frameworks for wireless ad hoc networks, in: IEEE infocom, Citeseer, 2004, pp. 1284-1295.
[52] C.E. Shannon, A mathematical theory of communication, ACM SIGMOBILE Mobile Comput. Commun. Rev. 5 (2001) 3-55.
[53] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1-30.
[54] Y.-Y. Chen, C. Chen, Simulated annealing for interface-constrained channel assignment in wireless mesh networks, Ad Hoc Networks, 29 (2015) 32-44.
[55] R.D. Vallam, A.A. Kanagasabapathy, C.S. Murthy, A non-cooperative game-theoretic approach to channel assignment in multi-channel multi-radio wireless networks, Wireless Networks, 17 (2011) 411-435.
[56] D. Yang, X. Fang, G. Xue, Channel allocation in non-cooperative multi-radio multi-channel wireless networks, in: 2012 Proceedings IEEE INFOCOM, 2012, pp. 882-890.