Connection Science
Vol. 00, No. 00, Month 200x, 1–19

RESEARCH ARTICLE

The Limits and Robustness of Reinforcement Learning in Lewis Signaling Games

David Catteeuw* and Bernard Manderick

Artificial Intelligence Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
(*Received 00 Month 200x; final version received 00 Month 200x*)

Lewis signaling games are a standard model to study the emergence of language. We introduce *win-stay/lose-inaction*, a random process that only updates behavior on success and never deviates from what was once successful, prove that it *always* ends up in a state of *optimal* communication in *all* Lewis signaling games, and predict the number of interactions it needs to do so: N^3 interactions for Lewis signaling games with N equiprobable types. We show three reinforcement learning algorithms (Roth-Erev learning, Q-learning, and Learning Automata) that can imitate win-stay/lose-inaction and can even cope with errors in Lewis signaling games.

1. Introduction

We study how repeated interaction between reinforcement learners may lead to optimal equilibria in Lewis signaling games (Lewis, 1969). These are two-player games of common interest where the players are successful once they reach an agreement on the meaning of some arbitrary signals. They model a communication problem and are the standard game theoretic model to study the emergence of signaling systems, such as language (Skyrms, 2010). We provide more details on Lewis signaling games in Section 2.

It is desirable to have a good understanding of the mechanisms by which efficient signaling systems may be established, since signaling (or communication) is an important aspect of our everyday life and that of all social organisms (D'Ettorre & Hughes, 2008). Examples of signaling in nature, include the alarm calls of Vervet monkeys (Seyfarth, Cheney, & Marler, 1980), cooperative hunting of fish (Bshary, Hohner, Ait-el Djoudi, & Fricke, 2006), the honey bees' waggle dance (Von Frisch, 1967), and signaling among bacteria (Dunny & Winans, 1999). Signaling also played a major role in the self-organization of biological systems (Maynard Smith & Szathmary, 1997). Our research provides evidence that meaning, and signaling systems as a whole, can emerge by chance through very simple adaptive dynamics, such as reinforcement learning.

The emergence of communication is also interesting from the viewpoint of artificial intelligence. The designer of a multiagent system usually implements a shared coordination protocol which allows the agents to coordinate (Tambe, 1997). Agents who do not share a common coordination protocol, may still coordinate if they share a common language (S. Barrett, Agmon, Hazon, Kraus, & Stone, 2013). If the agents can learn or bootstrap a language, the system designer does not even need to design the language beforehand—he may simply provide the agents with

*Corresponding author. Email: dcatteeu@vub.ac.be

ISSN: 0954-0091 print/ISSN 1360-0494 online

DOI: 10.1080/09540090xxxxxxxxxxxx
http://www.informaworld.com

the necessary learning capabilities. A language invented by the agents themselves may be more efficient than what a system designer can come up with, since the language will be adapted to the specific coordination problem the agents face and may even coevolve with the problem. Servin and Kudenko (2008) implemented a multiagent system to detect intrusions into a computer network. Since the system designers did not know what should be communicated, they allowed the agents to develop their own language.

The Lewis signaling game is a tough coordination game due to the existence of many suboptimal equilibria, so-called "pooling" or "partial pooling equilibria" (Section 2). Some algorithms easily get stuck in these suboptimal equilibria. Other algorithms, such as win-stay/lose-randomize, always lead to an optimal equilibrium in theory, but require too much time in practice for all but the smallest games (Section 5). A more acceptable solution to the problem of pooling equilibria should not introduce a random, but a guided drift towards more optimal signaling.

Our *first contribution* experimentally shows that Roth-Erev learning, Q-learning, and Learning Automata (Section 3) are able to overcome the problem of suboptimal equilibria and quickly find a state of optimal signaling (Section 4.1). We also explain why this is the case (Section 4.2). In extreme cases, these algorithms follow a heuristic we call "win-stay/lose-inaction." Just as win-stay/lose-randomize, win-stay/lose-inaction repeats behavior that is successful, but, whenever it is unsuccessful win-stay/lose-inaction does not alter its behavior (win-stay/lose-randomize would choose an action at random next time).

Our *second contribution* (Section 4.3), an analysis of win-stay/lose-inaction in Lewis signaling games, proves that a state of optimal communication is always reached and predicts the number of interactions needed to do so. For Lewis signaling games with uniform type[1] distributions, the expected number of interactions needed to find a convention is approximately N^3, where N is the size of the game: the number of types, signals, and responses. Experiments with reinforcement learning confirmed this result.

Our *third contribution* (Section 4.4) shows that Roth-Erev learning, Q-learning, and Learning Automata always learn to signal optimally even if errors occur, that is, they are robust. This is an important step toward their application in real world problems.

2. The Lewis Signaling Game

A Lewis signaling game (Lewis, 1969) is a two-player extensive form game such as the one shown in Figure 1. The first player is called *Sender*, the second is called *Receiver*. Sender has some information, his *type*, which Receiver cannot directly observe, but Sender can send a signal which Receiver can observe. It is in both players' interest that Receiver can reliably deduce Sender's type.

The game is played as follows. First, Nature determines Sender's type t_i by drawing t_i according to a discrete probability distribution π. The type is only known to Sender. Next, based on his type, Sender sends a signal m_j to Receiver, who, based on this signal, selects a response r_k. The number of types, signals, and responses is N: i, j, and $k \in \{1, 2, \ldots, N\}$. Finally, Sender and Receiver get a payoff

[1] In game theory, a type is an agent's private information. In Lewis signaling games the first agent should try to communicate his type to the other agent.

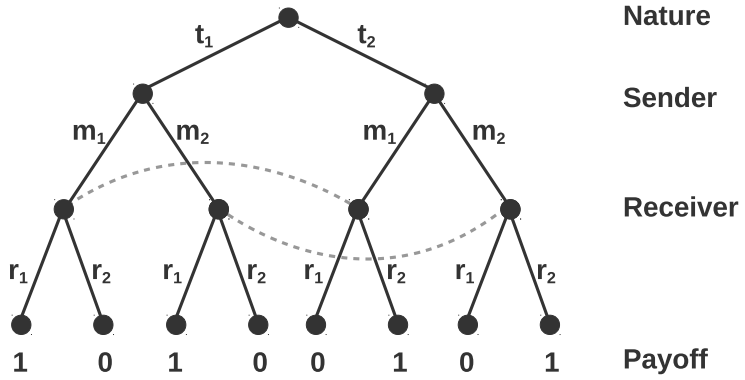


Figure 1. A Lewis signaling game with two types, signals, and responses, $N = 2$. The game is played as explained in the text. A dashed line indicates that the decision nodes it connects cannot be distinguish from each other.

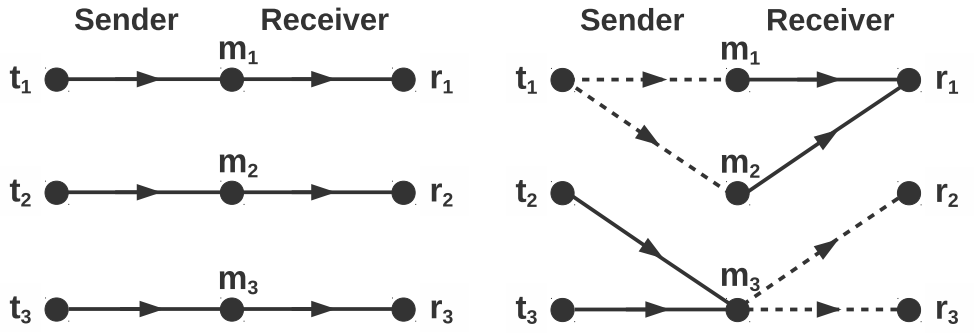
u depending on Sender's type t_i and Receiver's response r_k as follows:

$$u = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

For each type t_i, there is exactly one correct response r_i, and each response r_k is optimal for exactly one type t_k. The payoff u does not depend on the signal sent. Since the number of types, signals, and responses are equal, a Lewis signaling game is completely determined by its type distribution π. As usually in game theory, the agent cannot rely on the order of the types, signals, and responses even though we index them with numbers. Instead of Equation (1), we could as well use any other one-to-one correspondence between types and responses, and this would not change the game.

In essence: the agents face a communication problem, which they can solve by establishing a shared language or convention—that is to say, by adopting compatible mappings from types to signals (by Sender) and from signals to responses (by Receiver). Their mappings will be compatible if they, when applied one after the other (first Sender's, then Receiver's mapping), lead to the correct response for all types. For a Lewis signaling game with N types, signals, and responses there are $N!$ different but equally valid and optimal conventions, corresponding to the $N!$ unique mappings from types to signals, or from signals to responses. One such convention is shown in Figure 2(a) for $N = 3$. In this figure, the *signaling success rate*, which we define as the probability that the two agents will have a successful interaction, is 1. Since the payoff for success is 1, and the payoff for failure is 0, the expected payoff for following a shared convention also is 1. Lewis calls such an optimal equilibrium a *signaling system*.

A Lewis signaling game also has many other equilibria, which are suboptimal, especially when the number of types is larger than two ($N > 2$). Such equilibria are called "pooling" or "partial pooling equilibria", because they involve Sender using the same signal for different types. Figure 2(b) shows a partial pooling equilibrium where Sender uses signal m_3 both for type t_2 and t_3. When observing signal m_3, Receiver can only guess what is the true type of Sender. Assuming that all types are equally likely, the signaling success rate (and the expected payoff) for the agents is 2/3: type t_1 always yields success, type t_2 and t_3 yield success only half of the time. This state is an equilibrium, since, neither Sender nor Receiver can change his mapping to increase his payoff. Sender's mapping is a best response to Receiver's mapping, and vice versa, Receiver's mapping is a best response to



(a) An optimal equilibrium or signaling system. For each type t_i Sender sends a distinct signal m_i.

(b) A partial pooling equilibrium. For both types t_2 and t_3 Sender sends the same signal m_3.

Figure 2. Agent strategies for a signaling game with $N = 3$ types, signals, and responses. Sender's strategies map types t_i to signals m_j and Receiver's strategies map signals m_j to responses r_k. A solid line represents a probability of 1. A dashed line represents a probability of 1/2.

Sender's mapping. It is the existence of many such suboptimal equilibria that makes it hard to find an optimal one.

Another difficulty arises when the type distribution π is non-uniform. For example, if one of the types has a probability of 90%, Receiver can simply ignore the signals, can always pick the action corresponding to the most frequent state, and will already attain a 90% signaling success rate.

As mentioned in Section 1, some learning algorithms may get stuck in these suboptimal equilibria and never reach an optimal one. Others are guaranteed to find an optimal equilibrium in theory, but require too much time in practice. We will show that the algorithms of the next section are able to overcome both difficulties and find a convention fast.

3. Reinforcement Learning

We consider three reinforcement learning rules: Roth-Erev learning (Roth & Erev, 1995), Q-learning (Watkins, 1989), and Learning Automata (Narendra & Thathachar, 1989). We restrict the analysis to reinforcement learning algorithms, since we want to find the "simplest" adaptive mechanisms that can still efficiently lead to a signaling system. Reinforcement learning algorithms are simple in two aspects: they require little information and computational capacity. Roth-Erev learning and Q-learning are so-called action value methods. Such methods consist of an *update rule*; an *action selection rule*; and a value $q_{s,a}$, for each state-action pair (s, a) that indicates the quality of taking action a in state s relative to the other actions. The update rule determines how action values are updated based on new experience. The action selection rule determines which action to select, given the current state and the action values, by calculating the probability $p_{s,a}$ of taking action a in the current state s for all actions a. The usual constraints on probabilities hold ($\forall s : \sum_a p_{s,a} = 1$ and $\forall s, a : p_{s,a} \geq 0$). The basic idea is that action values of successful actions increase and actions with higher values get selected more often than actions with lower action values. Learning Automata are similar but directly update the probability distribution over the actions.

To apply these algorithms to signaling games, Sender will have an action value $q_{t,m}$ and a probability $p_{t,m}$, for all types t and signals m. Likewise, Receiver will have an action value $q_{m,r}$ and a probability $p_{m,r}$, for all signals m and responses r. So, signals take the role of actions for Sender and the role of states for Receiver.

Of course, Learning Automata only have probabilities $p_{s,a}$ for all state-action pairs (s, a) and no action values $q_{s,a}$.

Q-learning as introduced by Watkins (1989) is a temporal difference method—it is capable of crediting a reward to actions which were taken some time ago. Players do not need such capabilities in signaling games because they arrive at just one decision point (or state) and take only one action per game. The reward is easily credited to that action. We can use a simplified version of Q-learning: the so-called *single-state Q-learning*.

3.1. *Roth-Erev learning*

Roth-Erev learning (Roth & Erev, 1995) has two parameters: (a) a discount factor $\lambda \in [0, 1]$ and (b) an initial action value $q(0) \geq 0$. All action values are initialized to $q(0)$. Given the current state s, action a is selected with probability $p_{s,a}$ proportional to the action value $q_{s,a}$:

$$p_{s,a} = \frac{q_{s,a}}{\sum_{a'} q_{s,a'}} \tag{2}$$

This assumes that all action values (and consequently, payoffs) are non-negative ($\forall s, a : q_{s,a} \geq 0$). When all action values are 0, each action is selected with equal probability.

After taking action a and receiving payoff u, all action values for the current state s are discounted by factor λ and the action value of the current action is incremented with payoff u:

$$q_{s,a} \leftarrow \begin{cases} \lambda q_{s,a} + u & \text{if action } a \text{ was taken,} \\ \lambda q_{s,a} & \text{otherwise.} \end{cases} \tag{3}$$

Action values for states other than the current one are not updated.

In Roth and Erev's basic model the discount factor $\lambda = 1$ and the initial action value $q(0) = 1$. A small discount factor ($\lambda < 1$) helps forgetting old experience. It bounds the action values to $u/(1 - \lambda)$, which makes it possible for the algorithm to settle down. If the discount factor $\lambda = 1$, the action values are unbounded and the system never settles down. Using small initial action values ($q(0) < u$) speeds up learning in the beginning because it increases the importance of the payoffs u.

3.2. *Q-learning*

In Q-learning, all action values are initialized to $q(0) \in \mathbb{R}$. High initial action values increase the amount of exploration in the beginning of the learning process (Sutton & Barto, 1998, p39).

Q-learning can be combined with different action selection rules, like ϵ-greedy and softmax action selection. We describe and use both. With probability ϵ, *ϵ-greedy action selection* selects an action at random; and with probability $1 - \epsilon$ it selects the action with the highest action. In the latter case, if multiple actions have the maximum action value, these are selected with equal probability. If $\epsilon = 0$ then there is no exploration. We call this "greedy action selection". *Softmax action selection* selects an action a for current state s with probability $p_{s,a}$ according to

the Boltzmann distribution:

$$p_{s,a} = \frac{e^{(q_{s,a}/\tau)}}{\sum_{a'} e^{(q_{s,a'}/\tau)}}, \tag{4}$$

where temperature τ controls the rate of exploration: much exploration for high temperatures, little exploration for low temperatures. To let the behavior stabilize, the exploration rate ϵ or temperature τ can be decreased over time. We can decrease them fast: $\epsilon(i) = \min(\epsilon, \epsilon/i)$, or slow: $\epsilon(i) = \min(\epsilon, \epsilon \log(i)/i)$, where $\epsilon(i)$ is the exploration rate used at the ith iteration, and ϵ is the determined by the user. For softmax exploration, simply replace ϵ by τ.

After taking action a in state s (according to one of the action selection rules) and receiving payoff u, the action value $q_{s,a}$ is updated while the other action values remain unchanged:

$$q_{s,a} \leftarrow \begin{cases} q_{s,a} + \alpha(u - q_{s,a}) & \text{if action } a \text{ was taken,} \\ q_{s,a} & \text{otherwise,} \end{cases} \tag{5}$$

where $\alpha \in [0, 1]$ is the learning rate. A higher learning rate puts more weight on more recent payoffs. If α is 0, nothing is ever learned; if α is 1, the action value of an action simply equals the last payoff earned for that action.

The algorithm has three parameters: (a) a learning rate $\alpha \in [0, 1]$, (b) an initial action value $q(0) \in \mathbb{R}$, and (c) an exploration rate $\epsilon \in [0, 1]$ in the case of ϵ-greedy action selection, or a temperature $\tau > 0$ in the case of softmax action selection.

3.3. *Learning Automata*

Learning Automata (Narendra & Thathachar, 1989) directly update the probability distribution over the actions and have two parameters: (a) a reward factor $\alpha \in [0, 1]$ and (b) a penalty factor $\beta \in [0, 1]$. The probability distribution over the actions starts off uniform and changes as follows after receiving payoff u for taking an action in state s:

$$p_{s,a} \leftarrow \begin{cases} p_{s,a} + \alpha u(1 - p_{s,a}) - \beta(1 - u)p_{s,a} & \text{if action } a \text{ was taken,} \\ p_{s,a} - \alpha u p_{s,a} + \beta(1 - u)(\frac{1}{n-1} - p_{s,a}) & \text{otherwise,} \end{cases} \tag{6}$$

where n is the number of actions in state s. The learning rule requires that the payoff $u \in [0, 1]$ (which is the case for Lewis signaling games).

There are several well-known schemes, one is Linear-Reward-Inaction (L_{R-I}), another is Linear-Reward-ϵ-Penalty ($L_{R-\epsilon P}$). In Linear-Reward-Inaction, learning rate $\beta = 0$, and thus, it only updates the action probabilities on reward (payoff $u > 0$). In Linear-Reward-ϵ-Penalty, β is a fraction of α.

4. Results

4.1. *Experiments*

We now turn to some experimental results with these algorithms. We concentrate on two performance criteria: (a) whether or not a convention is reached and (b) the number of iterations needed to reach a convention. Figure 3 shows the results for the various algorithms and different parameters for the Lewis signaling game

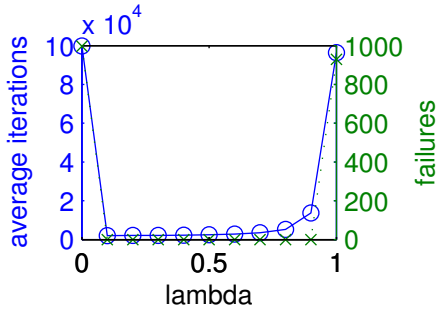
with type distribution $\pi = \langle \frac{1}{36}, \frac{2}{36}, \frac{3}{36}, \frac{4}{36}, \frac{5}{36}, \frac{6}{36}, \frac{7}{36}, \frac{8}{36} \rangle$. Per experiment, we did 1,000 runs. Per run, we recorded how many iterations were needed to reach a convention. If, after 100,000 iterations, still no convention was reached, we stopped the run and counted it as a failure.

We say that Sender and Receiver reach a convention if the signaling success rate is above some *threshold* θ. Remember that the signaling success rate is the probability that the next game will be successful. We do not set the threshold to $\theta = 1$. This would be too strict, since some strategies can never reach this level although they perform almost optimal. ϵ-greedy Q-learning for example can never achieve a signaling success rate of 1, unless the exploration rate $\epsilon = 0$. Therefore, we choose to set the threshold θ halfway between the optimal value, which is 1, and the signaling success rate of the best suboptimal equilibrium. This allows to clearly distinguish between runs which do not end up in an optimal state but are very near and runs that are closer to a suboptimal equilibrium than an optimal one.

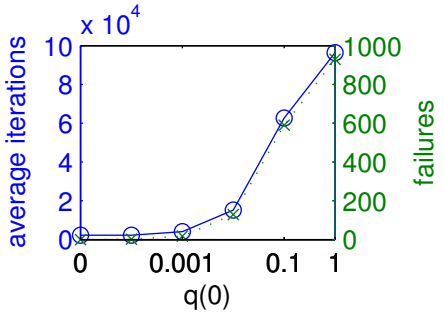
The *best suboptimal equilibrium* has a signaling success rate of 1 minus the probability of the rarest type: $1 - \min_t(\pi_t)$, where π_t is the probability that Nature draws type t. The best suboptimal equilibrium is actually a partial pooling equilibrium, such as the one in Figure 2(b), where Sender uses the same signal m_3 for different types. When seeing this signal m_3, Receiver cannot distinguish between the types t_2 and t_3, and hence can do no better than assuming the most frequent type. So, whenever *the other* type occurs, the game fails. In all other cases, the game succeeds. Thus, the signaling success rate in the *best* partial pooling equilibrium is determined by the frequency of the *rarest* type. For the example in Figure 3, the rarest type is $t_1 = \min_t(\pi_t)$ with probability $\pi_{t_1} = 1/36$. So, the threshold is at $\theta = 71/72$, halfway between 1 and 35/36.

The experiments tell us the following:

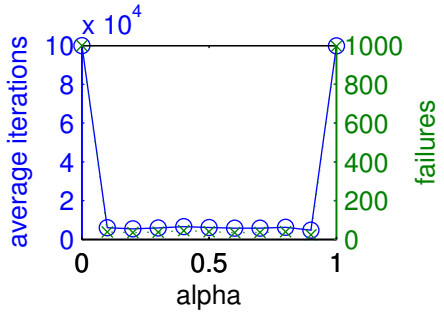
- Roth-Erev learning performed best with a small but positive discount-factor ($0 < \lambda \ll 1$, Figure 3(a)). Roth-Erev learning with $\lambda = 0$ is the same as win-stay/lose-randomize. Although, in theory, it always reaches a convention; in this example, it is too slow and none of the 1,000 runs reached a convention in less than 100,000 iterations. We reported this previously (Catteeuw, De Beule, & Manderick, 2011).
- Roth-Erev learning performed better when initial action values were very small and best when they were zero ($q(0) = 0$, Figure 3(b)). This was also reported by Skyrms (2010, p97).
- Q-learning performed well at any learning rate $0 < \alpha < 1$ (Figure 3(c)), and best when playing greedy (exploration rate $\epsilon = 0$, Figure 3(d)).
- Combining Q-learning with other action selection strategies also revealed that playing greedy works best. Q-learning with softmax action selection performed well at low temperatures τ. Both ϵ-greedy and softmax Q-learning performed well with decreasing exploration rates ϵ and temperatures τ. Decreasing these parameters fast worked better than decreasing them slowly. The rest of the text only discusses ϵ-greedy action selection.
- Learning Automata performed better for high reward rates α (Figure 3(e)) and low penalty rates β (Figure 3(f)). They performed best when the learning rate $\alpha = 1$ and $\beta = 0$.
- Finally, all three algorithms performed equally well in the best case. For optimal parameters, each algorithm needed slightly more than 2,000 iterations on average and always found a convention in less than 100,000 iterations. For other Lewis signaling games we found similar results. As you may expect, more types and non-uniform distributions required more



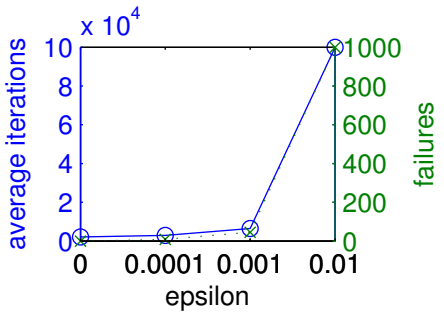
(a) Roth-Erev learning for different discount factors λ and initial action value $q(0) = 1$.



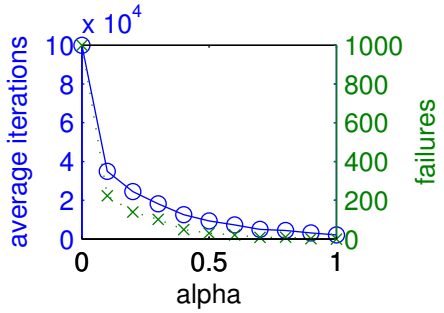
(b) Roth-Erev learning for different initial action values $q(0)$ and discount factor $\lambda = 1$.



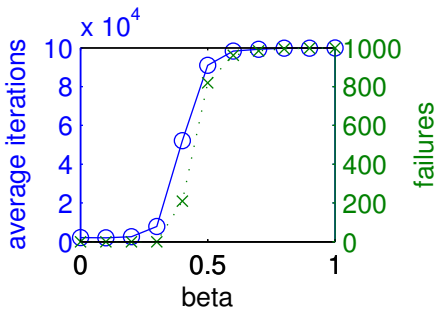
(c) ϵ-greedy Q-learning for different learning rates α and exploration rate $\epsilon = 0.001$.



(d) ϵ-greedy Q-learning for different exploration rates ϵ and learning rate $\alpha = 0.1$.



(e) Learning Automata for different reward factors α and penalty factor $\beta = 0$.



(f) Learning Automata for different penalty factors β and reward factor $\alpha = 1$.

Figure 3. The six figures show the number of iterations needed to learn a convention averaged over 1,000 runs (blue, solid lines with circles, left y-axis) and the number of runs (out of 1,000) that failed to find a convention in less than 100,000 iterations (green, dotted lines with crosses, right y-axis) for the Lewis signaling game with type distribution $\pi = \langle \frac{1}{36}, \frac{2}{36}, \frac{3}{36}, \frac{4}{36}, \frac{5}{36}, \frac{6}{36}, \frac{7}{36}, \frac{8}{36} \rangle$.

iterations.

For completeness, we mention some algorithms that yielded unsatisfactory results. UCB1 (Auer, Cesa-Bianchi, & Fischer, 2002) always found a convention but is a factor slower than greedy Q-learning. Some algorithms sometimes, or always, failed to find a convention. These are EXP3, EXP3.1, EXP3.S (Auer, Cesa-Bianchi, Freund, & Schapire, 2003), a Reinforcement Comparison technique (Sutton & Barto, 1998, ch2), and a Pursuit method (Sutton & Barto, 1998, ch2). In the remainder of the text we focus on Roth-Erev learning, (ϵ-greedy) Q-learning, and Learning Automata.

4.2. *Win-stay/lose-inaction*

We can explain the good performance of these three algorithms (Roth-Erev learning, Q-learning, and Learning Automata) by showing their similarity to a random process we call "win-stay/lose-inaction". Let us first introduce this simple process and explain how it behaves in the Lewis signaling game. Win-stay/lose-inaction works as follows:

- Initially, play random.
- Repeat forever the first action that yields success.

So, win-stay/lose-inaction never changes its behavior after a failure, whether this is due to an action that was randomly chosen or one that was previously successful.

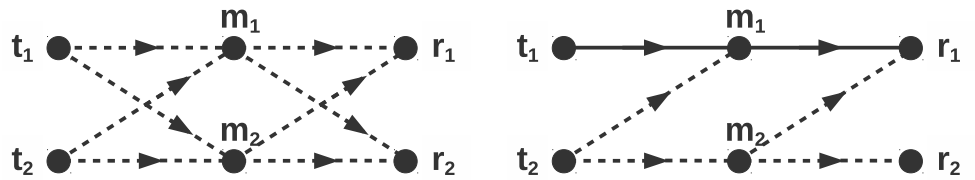
As an example, consider how win-stay/lose-inaction learns a convention in a Lewis signaling game with $N = 2$ types, signals, and responses. Figure 4(a) shows the initial behavior for both Sender (from types t to signals m) and Receiver (from signals m to responses r). Initial behavior is random and the behavior will remain this way until the first successful interaction. After the first success, we can relabel the successful type, signal, response as t_1, m_1, and r_1, respectively, without loss of generality. Remember that the ordering of types, signals, and responses is of no concern to the agents. From now on, Sender will always use signal m_1 when observing type t_1 and Receiver will always respond with r_1 to signal m_1. We say a path $t_1 \rightarrow m_1 \rightarrow r_1$ is learned for type t_1. Behavior for other types and signals remains random (Figure 4(b)). Several things can happen now.

(1) Nature draws type t_1. This will trigger Sender to send signal m_1, and consequently, Receiver will respond with r_1. So, this will always lead to success, and the path $t_1 \rightarrow m_1 \rightarrow r_1$ persists. More generally, whenever a type occurs for which a path was already learned, the interaction is successful and the agents do not change their behavior.
(2) Nature can also draw t_2, or, more generally, a type for which no path is yet learned. Sender will signal at random and there are again two possibilities:
    a) If he picks signal m_1 (or more generally, a signal which is already used in a learned path), then Receiver will definitely respond with r_1 and the game fails. The agents do not update their behavior in that case.
    b) If, on the other hand, Sender chooses m_2, (or in general, any signal which is not yet used in a path), then either Receiver guesses the incorrect response, or he guesses the correct response. In the former case, the behavior of both agents remains the same. In the latter case, they will update their behavior and a new path $t_2 \rightarrow m_2 \rightarrow r_2$ will complete the convention.

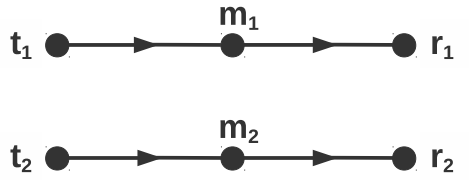
Each interaction will now be successful (Figure 4(c)). It is straightforward to generalize the reasoning above to all Lewis signaling games. Behavior changes only if (a) Nature selects a type t_i that does not yet have a learned path, (b) Sender selects a signal m_j that is not yet used in a learned path, and (c) Receiver selects the correct response r_i. In that case, a new path is learned for type t_i.

The three reinforcement learning rules from Section 3, behave like win-stay/lose-inaction. Learning Automata do so when the learning rate $\alpha = 1$ and $\beta = 0$. First, initial behavior is random. Second, when an interaction fails (payoff $u = 0$), the probabilities over the actions do not change (Equation (6)). Third, when an interaction succeeds (payoff $u = 1$), the probability of the action taken, becomes 1, and all others 0, independent of the current probabilities.

Roth-Erev learning behaves like win-stay/lose-inaction when the initial action values $q(0) = 0$ and the discount factor $0 < \lambda \leq 1$. Q-learning behaves like win-



(a) Initial behavior is entirely random. *(b)* Path $t_1 \rightarrow m_1 \rightarrow r_1$ was learned.



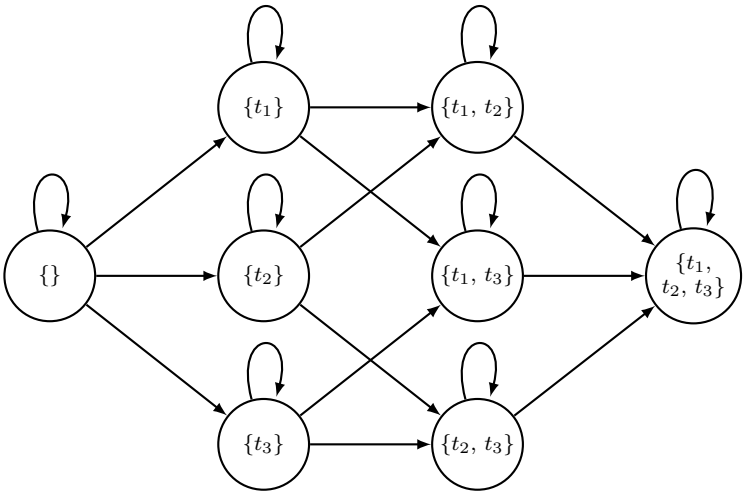
(c) Final mapping.

Figure 4. Evolution of learning by win-stay/lose-inaction for a Lewis signaling game with $N = 2$ types. A solid line represents a probability of 1. A dashed line represents a probability of $1/N$.

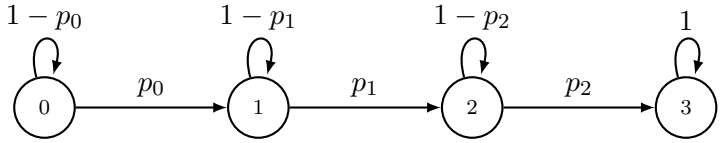
stay/lose-inaction when the initial action values $q(0) = 0$, the learning rate $0 < \alpha < 1$, and the exploration rate $\epsilon = 0$. For Roth-Erev and Q-learning, it is somewhat harder to see this, because the action values do *change on failure* (the action value of the action resulting in failure is slightly decreased), but it would take an infinite number of failures before the probability distribution over the actions also changes.

4.3. *Number of interactions needed to reach convention*

Now that we understand how win-stay/lose-inaction behaves in Lewis signaling games we can actually prove that it always reaches a convention and predict the number of iterations needed, on average, to do so. We therefore map the learning process onto the Markov chain that has a state for each possible subset of the set of types. This subset of types represents the types for which a path has already been learned. Figure 5(a) shows the Markov chain for Lewis signaling games with two types ($N = 3$). There is one initial state where none of the types has a path. There is also one final state where all types have a path. Furthermore, there are N states where one type has a path, etc. Since, win-stay/lose-inaction can never forget a learned path, the Markov chain can never get into a state with fewer learned paths than the current state. For the same reason, the random process either remains in the same state, or it goes to a state where one extra path is learned. The probability to go to a next state is the probability that a new path is learned. It is the probability that (a) Nature selects a type for which no path yet exists, (b) Sender selects a signal that is not yet used in a path, and (c) Receiver selects the correct response. The probability that Nature selects a type for which no path yet exists is the sum of the individual probabilities of these types. The probability that Sender selects an unused signal is $\frac{N-l}{N}$, where N is the number of signals and l is the number of learned paths and used signals. The probability of selecting the correct response is $1/N$, where N is the number of responses. The product of these probabilities is the probability of learning a new path. This probability is different in each state of the Markov chain. Note that this Markov chain is absorbing since (a) there is an absorbing state and (b) there is a path with positive probability from each state to the absorbing state. Since furthermore, the absorbing state is unique, we always end up in that state which is also the state of optimal communication. This proves that win-stay/lose-inaction always ends up in a state of optimal communication.



(a) General type distributions. Each state of the Markov chain is represented by the set of types for which the game is always successful.



(b) Uniform type distributions. Each state of the Markov chain is represented by the number of types for which the game is always successful. The transition probabilities p_l for $l = 0$, 1, and 2 can be computed from Equation (7).

Figure 5. The Markov chains for Lewis signaling games with $N = 3$ types.

For Lewis signaling games with uniform type distributions and N types, the Markov chain can be simplified, since all types have the same probability. The *number of types* for which a path is learned is sufficient to discriminate all states of the Markov chain (Figure 5(b)). In that case, the probability of selecting a type for which no path yet exists is $\frac{N-l}{N}$, where N is the number of types and l the number of learned paths. So, the probability of learning a new type-signal-response path is

$$p_l = \frac{N-l}{N}\frac{N-l}{N}\frac{1}{N} = \frac{(N-l)^2}{N^3}. \tag{7}$$

This probability p_l corresponds to the second part of case 2b of the analysis in Section 4.2. In all other cases, the Markov process remains in the same state. This happens with probability $1 - p_l$.

The expected number of iterations to learn a new path is distributed according to a geometric distribution which has mean $\mu = 1/p$, where p is the probability to learn a new path. The expected number of iterations $E[T_c]$ to learn a complete mapping for a Lewis signaling game with N types and uniform type distribution is the sum of the expected number of iterations needed for each new path:

$$E[T_c] = \sum_{l=0}^{N-1} \frac{1}{p_l} = \sum_{l=0}^{N-1} \frac{N^3}{(N-l)^2} = N^3 \sum_{i=1}^{N} \frac{1}{i^2}. \tag{8}$$

The sum $\sum_{i=1}^{N} \frac{1}{i^2}$ slowly converges to $\pi^2/6 \approx 1.64$ while N goes to infinity (Daners, 2012).

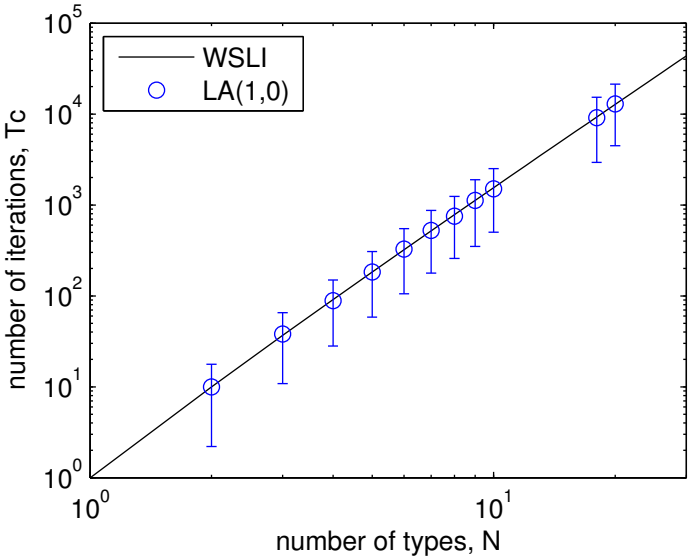


Figure 6. Theory predicts the number of iterations Learning Automata need to find a convention in Lewis signaling games with uniform type distributions. The solid black line shows what theory predicts for win-stay/lose-inaction. The blue circles show the average time and the error bars show the standard deviation over 1,000 simulations of Learning Automata with learning rate $\alpha = 1$ and $\beta = 0$ (LA(1,0)).

Equation (8) matches the data we generated by applying the three reinforcement learning rules to Lewis signaling games with different number of types and uniform type distributions. Figure 6 shows the results for (Linear-Reward-Inaction) Learning Automata with learning rate $\alpha = 1$ and $\beta = 0$. Regression analysis[1] on this data predicts that the number of iterations needed before reaching optimal signaling is $E[T_c] \approx 1.24N^{3.09}$.

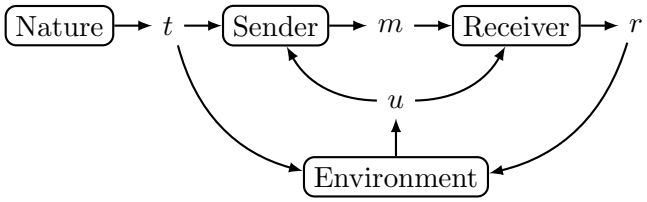
In a similar way one can calculate the expected time $E[T_c]$ needed to reach a convention in Lewis signaling games with general type distributions. We cannot give a general formula as the probability to learn a new path now depends on the types (not just the number of types) for which paths are already learned. This allows for multiple trajectories through the Markov chain (Figure 5(a)), one per possible way of ordering the types. That is, $N!$ in total. We can, however, provide a lower and upper bound. For non-uniform type distributions, the probability to learn a new type-signal-response path p_l is bounded from below and above:

$$\frac{(N-l)^2}{N^2} \min_t(\pi_t) \leq p_l < \frac{(N-l)^2}{N^3},$$

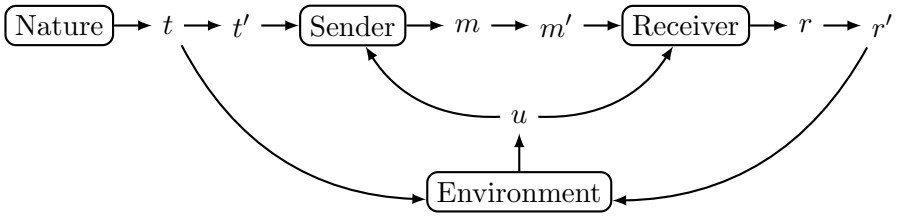
where $min_t(\pi_t)$ is the probability of the rarest type. Substituting this in Equation (8), we find a lower and upper bound for the expected number of iterations until we reach an optimal equilibrium:

$$N^3 \sum_{i=1}^{N} \frac{1}{i^2} < E[T_c] \leq \frac{N^2}{\min_t(\pi_t)} \sum_{i=1}^{N} \frac{1}{i^2}.$$

[1]We performed a standard linear regression analysis in log-log scale. The logarithm of the expected number of iterations is a linear function of the logarithm of the size of the game: $\log E[T_c] = A + B \log N$, where A and B are output of the regression analysis. Converting the result back to linear scale yields: $E[T_c] = 10^A N^B$.



(a) Standard Lewis signaling game.



(b) Lewis signaling game with errors.

Figure 7. Flow diagrams representing Lewis signaling games. See Section 2 for a full explanation of the game.

4.4. *Robustness to errors*

The results of our final experiments show that the reinforcement learning algorithms are robust to errors in the Lewis signaling game. We allow for three types of errors:

(a) the type t' observed by Sender may be different from the true type t,
(b) the signal m' observed by Receiver may be different from Sender's intended signal m, and
(c) the true response r' may be different from Receiver's intended response r.

The flow diagrams in Figure 7 illustrate the difference between the original game and the game with errors. Each of these errors occur with a (small) fixed probability p_e, which we call the *error rate*. In $(1 - p_e)^3$ interactions no errors occur.

An algorithm which is robust to errors should avoid getting locked in. Unfortunately, this is a key characteristic of win-stay/lose-inaction and hence it is not robust to errors. Whenever an error occurs and the interaction fails, no harm is done since win-stay/lose-inaction does not update its behavior on failure. Whenever an error occurs and the interaction succeeds, chances are win-stay/lose-inaction learns the wrong thing and will forever repeat its unsuccessful behavior. Consider the following example that leads to success, but where one error occurs. Nature selects t_1 as Sender's true type but Sender wrongfully observes t_2. Next, Sender selects signal m_2 as response to t_2 and Receiver responds with r_1 to signal m_2. Since the true type t_1 and the true response r_1 match, the interaction succeeds. The agents have now learned the path $t_2 \to m_2 \to r_1$ which fails under normal circumstances. Win-stay/lose-inaction can also learn correct behavior even though an error occurs, namely when errors "cancel out".

Win-stay/lose-inaction is only an idealized version of the reinforcement learning algorithms that allowed us to explain, prove, and predict their performance. Starting from the parameters that were most successful, we can slightly modify the three reinforcement learning algorithms such that they no longer get locked into whatever is first successful, and so, make them robust to errors. A positive initial action value ($q(0) > 0$) does that for Roth-Erev learning and Q-learning. A positive penalty factor ($\beta > 0$) does that for Learning Automata. Figure 8 shows the performance of these robust algorithms in Lewis signaling games with uniform

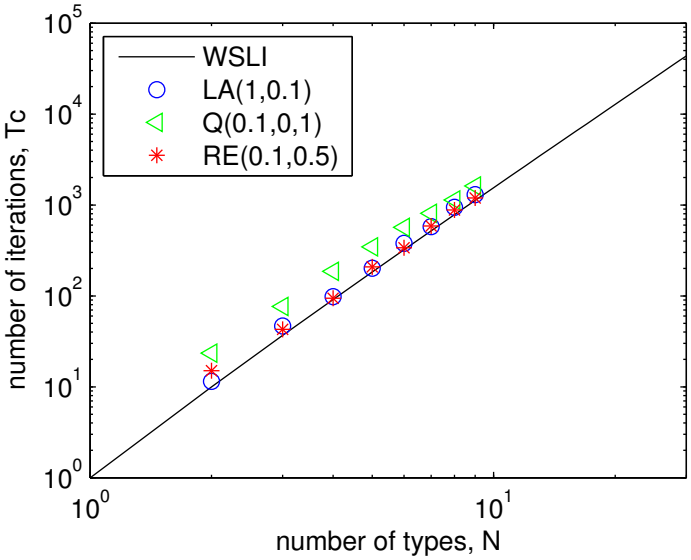


Figure 8. With error rate $p_e = 0.01$, the robust algorithms perform almost as well as theory predicts for win-stay/lose-inaction without errors. The data show the average number of iterations needed over 100 runs for Lewis signaling games with uniform type distributions of different size $N = 2, 3, \ldots, 9$. The figure shows Roth-Erev learning with discount factor $\lambda = 0.1$ and initial action value $q(0) = 0.5$; Q-learning with learning rate $\alpha = 0.1$, exploration rate $\epsilon = 0$, and initial action value $q(0) = 1$; Learning Automata with learning rate $\alpha = 1$ and $\beta = 0.1$.

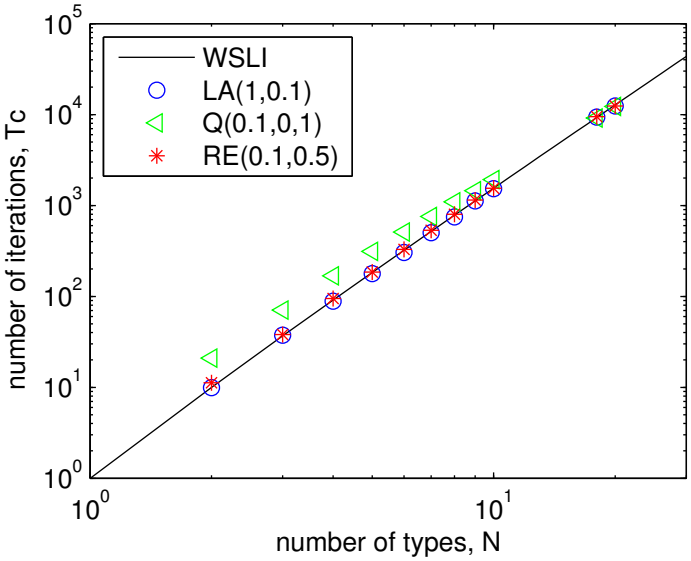


Figure 9. Without errors, the robust algorithms perform equally well as theory predicts for win-stay/lose-inaction. The data show the average number of iterations needed over 1,000 runs for Lewis signaling games with uniform type distributions of different size $N = 2, 3, \ldots, 9, 10, 18, 20$. Q-learning performed slightly worse for small games ($N \leq 10$), for larger games Q-learning performs at least as good as win-stay/lose-inaction. The algorithms used the same parameters as in Figure 8.

type distributions and error rate $p_e = 0.01$.

The learning algorithms are now capable of unlearning previous experience but this does not decrease their performance in Lewis signaling games without errors (Figure 9). Previously successful behavior is only forgotten after some consecutive failures. Under small error rates, truly successful behavior will rarely lead to failure and even more rarely to a sequence of failures, and thus will never be forgotten. Behavior that was only successful due to errors will soon yield a sequence of failures and be forgotten.

5. Related Work

5.1. *Reinforcement learning in Lewis signaling games*

Argiento et al. (Argiento, Pemantle, Skyrms, & Volkov, 2009) have proven that basic *Roth-Erev learning* (with initial action values $q(0) = 1$ and discount factor $\lambda = 1$) always converges to a signaling system in games with two equiprobable types ($\pi = \langle \frac{1}{2}, \frac{1}{2} \rangle$). Basic Roth-Erev learning fails whenever the number of types $N > 2$ or when the probability distribution π over the types is non-uniform (J. A. Barrett, 2006; Catteeuw et al., 2011; Huttegger, 2007). Skyrms (2010, p97) reported that smaller initial action values ($q(0) < 1$) increase the probability of reaching a signaling system, even when the number of types is larger than two ($N > 2$) and the probability distribution over the types π is non-uniform. Here, we were able to give an explanation of why this is the case. J. A. Barrett and Zollman (2009) apply win-stay/lose-randomize to Lewis signaling games and prove that it always reaches a signaling system. Win-stay/lose-randomize is equal to Roth-Erev learning when the discount factor $\lambda = 0$. Although this is theoretically very interesting, experiments show that the number of interactions needed to reach a signaling system increases exponentially with the size of the game (Catteeuw et al., 2011). Catteeuw et al. (2011) show experimentally how Roth-Erev learning with a discount factor $0 < \lambda < 1$ always reaches a signaling system and is much faster than win-stay/lose-randomize. J. A. Barrett (2006) studied two other variations of Roth-Erev learning for signaling games with an arbitrary number of types ($N \geq 2$) but uniform type distributions. One variation allows for negative rewards, the other randomizes action values. Both variations seem to help reaching a signaling system, but do not guarantee it.

J. A. Barrett and Zollman (2009) also discuss *softmax Q-learning* (but call it "smoothed reinforcement learning") and a more complex learning rule called "*ARP*" (Bereby-Meyer & Erev, 1998). They conclude that forgetting old experience increases chances of finding a convention. We found that this is unnecessary except for Lewis signaling games where errors occur.

Win-stay/lose-shift (which Skyrms (2010) calls "best response") does not work in Lewis signaling games with $N = 2$ types, since the process may get into endless loops. When there are more than two types ($N > 2$), the process must be redefined. For example, when loosing, we could pick any of the alternatives at random with equal probability. Skyrms calls this process "best response for all we know" (Skyrms, 2010, pp103-105). Still, this process cannot handle the case for $N = 2$ types, and Zollman proposes to add inertia: only now and then apply the best response rule. The rest of the time behavior is not updated.

5.2. *Steels language games*

The emergence of signaling systems (or a shared lexicon between multiple individuals) has been extensively studied before in the domain of language evolution by means of language games, see for example (Steels, 2001).

The guessing game (De Beule, De Vylder, & Belpaeme, 2006) is probably the language game which resembles the Lewis signaling game the most. It is played by two agents. The first is called Speaker, the second Hearer. At each interaction Speaker and Hearer are presented with a set of objects. This is called the *context* and varies from interaction to interaction. First, Speaker utters a word referring to one of the objects. Next, Hearer guesses which object Speaker refers to. Optionally, Speaker gives feedback to Hearer indicating whether or not he pointed to the correct object. In the guessing game, such feedback is indeed optional and is not at all

necessary for learning. Since the context changes from interaction to interaction, Hearer may infer from that context what a word can and cannot mean. The notion of a context is typical for language games and is the most important difference between language games and signaling games. Another difference: in signaling games, there is a one-to-one mapping between types and responses, which is not transparent to the agents; in language games, types and responses are both the same thing: objects. It is assumed that the agents know this and they are allowed to exploit this knowledge. This requires learning algorithms with some form of bi-directional memory: given a meaning, find the preferred signal; and given a signal, find the preferred meaning. Our work shows that simpler learning mechanisms, that do not assume a mapping between types and responses, can still lead to signaling systems.

5.3. *Coordination and Communication in Multiagent Systems*

The ability to communicate simplifies coordination and cooperation between artificial agents, as it allows agents to share observations and solutions to subproblems (Tan, 1993). But, communication also makes the problem harder, since agents must decide what to communicate when to whom.

Standardized communication protocols have been developed for this purpose, such as the Knowledge Query and Manipulation Language (or KQML) (Finin, Fritzson, McKay, & McEntire, 1994). The ability to bootstrap or adapt a communication system to the specific needs of the agent society allows for more flexibility and may frees the system designer from finding the delicate balance between a communication system which is sufficiently flexible, but does not impede learning (Jim & Giles, 2000; Servin & Kudenko, 2008; Yanco & Stein, 1993). Servin and Kudenko (2008) implemented a multiagent system where signals acquire their meaning exnihilo, adapted to the specific needs of the problem at hand. In this case, the agents are cooperating in order to detect intrusions into a computer network. Several sensor agents are spread throughout a network to monitor local traffic. They send signals to a decision agent, who—based on the signals of all sensor agents—decides whether or not to inform the network operator of a possible network intrusion. The key idea is that the sensor agents learn when to send what signal, while the decision agent learns to interpret the signals. The meaning of the signal is thus determined through learning and not determined beforehand by the system designers.

6. Conclusion

In order to gain more insight into the emergence of communication and more particularly the capabilities of reinforcement learning to bootstrap a communication system, we studied reinforcement learning in Lewis signaling games.

We have three main results. First, we found three reinforcement learning algorithms that always find a convention and do this reasonably fast. These algorithms are Learning Automata with learning rate $\alpha = 1$ and $\beta = 0$; greedy Q-learning with initial action value $q(0) = 0$ and learning rate $0 < \alpha < 1$; and Roth-Erev learning with initial action value $q(0) = 0$ and learning rate $0 < \lambda \leq 1$. These results hold for all Lewis signaling games, even those with more than two types ($N > 2$) and non-uniform type distributions. Second, we introduced the random process win-stay/lose-inaction which behaves exactly the same as these algorithms in Lewis signaling games. Markov chain analysis of win-stay/lose-inaction proves that it always reaches a convention and predicts the expected number of iterations before it does. The expected number of iterations needed is polynomial in the

size of the game: $N^3 \sum_{i=1}^{N} 1/i^2 \leq E[T_c] \leq (N^2/\min_t(\pi_t)) \sum_{i=1}^{N} 1/i^2$. Experiments confirmed this for the three reinforcement learning algorithms. Finally, slight adaptations render these algorithms robust without decreasing their performance in the original Lewis signaling games. The resulting algorithms are: Learning Automata with learning rate $\alpha = 1$ and positive $\beta > 0$; greedy Q-learning with positive initial action value $q(0) > 0$ and learning rate $0 < \alpha < 1$; and Roth-Erev learning with positive initial action value $q(0) > 0$ and learning rate $0 < \lambda \leq 1$.

These results are important from two points of view. First, it tells us that under very weak assumptions communication can emerge and can do this reasonably fast. The number of interactions necessary to reach optimal communication is polynomial in the number types, signals, and responses (N). In the meanwhile, more and more interactions will be successful. For really large communication systems, one would prefer time grows linear or even sublinear with the number of types N, and so other solutions are necessary. For example, if an agent could know which signal he uses for which type, he could avoid using that same signal for other types. This would definitely speed up learning, but it imposes extra requirements on the capabilities of the agents. This is exactly what researchers in the domain of language games are doing. Second, this result is also interesting for multiagent system designers, who now have an idea of what kind of learning rules work well and how many interactions such agents need to find a convention.

Lewis signaling games can be extended to more complex communication problems: feedback loops, where the response of Receiver influences Sender's type at the next iteration; dialogues, where the game consists of two stages, and the players' roles are reversed, games with multiple receivers, etc. (Skyrms, 2010, ch13). Future work could investigate whether simple learning mechanisms, such as reinforcement learning, can lead to efficient signaling in these more complex problems.

References

Argiento, R., Pemantle, R., Skyrms, B., & Volkov, S. (2009, February). Learning to signal: Analysis of a micro-level reinforcement model. *Stochastic Processes and their Applications*, *119*(2), 373–390.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, *47*(2), 235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2003). The Nonstochastic Multiarmed Bandit Problem. *SIAM Journal on Computing*, *32*(1), 48–77.

Barrett, J. A. (2006). *Numerical Simulations of the Lewis Signaling Game: Learning Strategies, Pooling Equilibria, and the Evolution of Grammar* (Tech. Rep. No. September). University of California, Irvine: Institute for Mathematical Behavioral Science.

Barrett, J. A., & Zollman, K. J. S. (2009, December). The role of forgetting in the evolution and learning of language. *Journal of Experimental & Theoretical Artificial Intelligence*, *21*(4), 293–309.

Barrett, S., Agmon, N., Hazon, N., Kraus, S., & Stone, P. (2013). Communicating with Unknown Teammates. In S. Devlin, D. Hennes, & E. Howly (Eds.), *Proceedings of the 13th adaptive and learning agents workshop* (pp. 46–52). Saint Paul, MN, USA.

Bereby-Meyer, Y., & Erev, I. (1998, June). On Learning To Become a Successful Loser: A Comparison of Alternative Abstractions of Learning Processes in the Loss Domain. *Journal of mathematical psychology*, *42*(2/3), 266–286.

Bshary, R., Hohner, A., Ait-el Djoudi, K., & Fricke, H. (2006, December). Inter-

specific communicative and coordinated hunting between groupers and giant moray eels in the Red Sea. *PLoS biology*, *4*(12), e431.

Catteeuw, D., De Beule, J., & Manderick, B. (2011). Roth-Erev Learning in Signaling and Language Games. In P. De Causmaecker, J. Maervoet, T. Messelis, K. Verbeeck, & T. Vermeulen (Eds.), *Proceedings of the 23rd benelux conference on artificial intelligence* (pp. 65–74). Ghent, Belgium.

Daners, D. (2012). A Short Elementary Proof of Σ 1/k2 = π2/6. *Mathematics Magazine*, *85*(5), 361–364.

De Beule, J., De Vylder, B., & Belpaeme, T. (2006). A cross-situational learning algorithm for damping homonymy in the guessing game. In L. M. Rocha, L. S. Yaeger, M. A. Bedeau, D. Floreano, R. L. Goldstone, & A. Vespignani (Eds.), *Proceedings of the tenth international conference on the simulation and synthesis of living systems* (pp. 466–472). MIT Press.

D'Ettorre, P., & Hughes, D. P. (2008). *Sociobiology of communication*. New York, NY, USA: Oxford University Press.

Dunny, G. M., & Winans, S. C. (1999). *Cell-cell signaling in bacteria*. American Society for Microbiology.

Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994). KQML as an agent communication language. In *Proceedings of the 3rd international conference on information and knowledge management* (pp. 456–463).

Huttegger, S. M. (2007, January). Evolution and the Explanation of Meaning. *Philosophy of Science*, *74*(1), 1–27.

Jim, K. C., & Giles, C. L. (2000, January). Talking helps: evolving communicating agents for the predator-prey pursuit problem. *Artificial life*, *6*(3), 237–54.

Lewis, D. K. (1969). *Convention: A Philosophical Study*. Cambridge: Harvard University Press.

Maynard Smith, J., & Szathmary, E. (1997). *The major transitions in evolution*. Oxford University Press.

Narendra, K. S., & Thathachar, M. A. L. (1989). *Learning automata: an introduction*. Upper Saddle River, NJ, USA: Prentice-Hall.

Roth, A. E., & Erev, I. (1995). Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, *8*(1), 164–212.

Servin, A. L., & Kudenko, D. (2008). Multi-Agent Reinforcement Learning for Intrusion Detection: A Case Study and Evaluation. *Lecture Notes in Artificial Intelligence, 6th German Conference on Multi-Agent System Technologies*, *5244*, 159–170.

Seyfarth, R. M., Cheney, D. L., & Marler, P. (1980, November). Vervet monkey alarm calls: Semantic communication in a free-ranging primate. *Animal Behaviour*, *28*(4), 1070–1094.

Skyrms, B. (2010). *Signaling: Evolution, Learning and Information*. Oxford University Press, New York.

Steels, L. (2001). Language Games for Autonomous Robots. *IEEE Intelligent Systems*, *September*, 17–22.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tambe, M. (1997). Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*, *7*, 83–124.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th international conference on machine learning*.

Von Frisch, K. (1967). *The dance language and orientation of bees*. Harvard

University Press.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards* (PhD). Cambridge University.

Yanco, H., & Stein, L. A. (1993). An Adaptive Communication Protocol for Cooperating Mobile Robots. In J. A. Meyer, H. L. Roitblat, & S. Wilson (Eds.), *From animals to animats 2. proceedings of the 2nd international conference on simulation of adaptive behavior* (pp. 478–485). Cambridge, MA: MIT Press.