SLA negotiation: Experimental Observations on Learning Policies

Invited Paper

Mohamed Lamine Lamali*, Hélia Pouyllau*, Dominique Barth†
*Alcatel-Lucent Bell Labs France. Route de Villejust, 91620 Nozay. France.
{mohamed_lamine.lamali, helia.pouyllau}@alcatel-lucent.com
†Laboratoire PR*i*SM. University of Versailles, 78000 Versailles. France.
dominique.barth@prism.uvsq.fr

***Abstract*—The Internet has moved to content broadcasting and one might anticipate future evolutions of the supported applications. Meanwhile, the Internet business model remains the same from the early-days. While, on the technological side, many discussions assess the ossification of the Internet, the Internet traded good is still reachability. Some authors argue that the technical ossification is a consequence of the economic one. But adopting a clean-slate economic model is as challenging as adopting a clean-slate architecture. The new system must meet requirements on profitability and stability while tackling complex issues. In this paper, we focus on the proposal of enriching Service Level Agreements (SLAs), which are contracts among Network Service Providers (NSPs) with Quality of Service (QoS) information. We propose a game model of the SLA negotiation among NSPs in order to study how some learning algorithms converge to stable conditions, which are mixed Nash Equilibria in this case. Computing a mixed Nash equilibrium is PPAD-complete; the corresponding algorithms are thus quite complex. In previous works, some authors studied the convergence of Reinforcement Learning techniques to pure and mixed Nash Equilibria. Learning mixed Nash Equilibria seems harder. Hence, we rather experimentally observe how such algorithms can, according to different policies, converge to mixed Nash Equilibria, and also how profitable they are for the NSPs.**

I. Introduction

In the recent years, the Internet has moved from a reachability-centric network to content broadcasting one [1]. This evolution led many authors to reconsider the business relations among Internet Network Service Providers (NSPs, also known as Autonomous Systems, ASes): in [2], Barth & al. proposed a learning scheme to capture optimal transit pricing, in [3], Ma & al. plead for the use of Shapley-value mechanisms to increase efficiency in revenue sharing, Valancius & al[4] argue for an open transit market.

The emergence of real-time applications (Cloud Computing, Gaming, etc.) and the anticipation of reliability-requiring one (e.g. telemedicine) demand *Quality of Services* (QoS) guarantees (e.g. delay ≤ 100 ms) that caching solutions (i.e. proposed by Content Delivery Networks, CDNs) cannot handle because of the best-effort nature of the Internet. Hence, to support QoS-guaranteed traffic new economic agreements must be considered. Existing business relationships in the Internet are numerous but derive from two kinds of relationships: public peering (i.e. two ASes interconnect without compensations) and transit (i.e. one AS pays the other). And the Internet's traded good is host reachability. Hence, to support inter-NSPs QoS based service, the agreements should move to Service Level Agreements (SLAs) embedding QoS promises (i.e. thresholds over various QoS parameters) and economic information (price, penalties, etc.).

From a game theory perspective, in the existing Internet, NSPs are large, independent and selfish agents, competing for resource usages (either end-user or content provider traffic). Enlarging this game to QoS, namely the *inter-NSP SLA negotiation* game, the NSPs would compete on the QoS they are able to propose and on the prices. In [5], the relationship with the customers is handled by a neutral third-party but in the general case, such relation could also be handled in a distributed manner. In both cases, the goal is to build an SLA chain which meets the customer utility based on his QoS requirements and willingness to pay.

In the SLA negotiation game, pure Nash Equilibria might not exist. Hence, to converge to mixed Nash Equilibria and to maximize both the customer and NSP welfare, we aim to experimentally investigate the convergence properties of various learning techniques in order to further analyze which one would fit the best the problem of SLA negotiation.

The paper is organized as follows: section II provided a summarized analysis of the existing Internet's business model and why it should evolve; section III gives a formulation of the SLA negotiation problem as a game; section IV focuses on Reinforcement Learning techniques and provides a description of the studied algorithms whose experimentation results are reported in section V.

II. From best-effort to QoS, challenges of the Future Internet

The Internet has been built upon the *end-to-end design principle*, which means that the control of flows operates at the end-points and thus that intermediate network nodes do not intervene. This lies to the set-up of best-effort policies in networks: the packets are routed according to the best that the network capacity can offer.

VALUETOOLS 2012, October 09-12, Cargèse, France

DOI 10.4108/valuetools.2012.250344

A. The limit of the existing model

As argued by Ben Houidi & al [6], the Internet traded good, namely the **reachability** is a legacy from the early-days Internet and suffers today from its vagueness. The reachability is solely defined as the capacity to reach a certain number of hosts. Under the major evolution of the Internet to content broadcasting [1], it is becoming more and more inadequate as contents move while end-users (early-day's Internet hosts) do not.

Furthermore, reachability is not charged in the same manner among NSPs and among NSPs and end-users. Among NSPs, the charging is volume-based while among NSPs and end-users it is flat-rate. This led to an unbalanced situation where end-users perceive incentives to consume more and more [7] while NSPs do not perceive incentives to invest in the network capacity.

To figure out this issue, many authors argued to change the Internet traded good using Quality of Service according to the congestion [8], [9], or application-based charging [10]. As noticed by Ben Houidi & al [6], changing the traded good is a partial action to "de-ossify" the Internet business model. Another key dimension lies in the bilateral nature of the agreements among NSPs. In this paper, we do not address this later issue. We focus on the exchange of new QoS goods among NSPs and intend to observe how learning algorithm can meet some of the corresponding challenges.

B. Challenges in SLA negotiation

The Internet routing protocol, called the Border Gateway Protocol (BGP) and standardized by the Internet Engineering Task Force (IETF), is also a legacy of the early-days Internet. Each NSP configures its BGP policy according to the existing agreements. This explains why the many standardization proposals to extend BGP with further information than reachability failed to convinced the IETF. Another important property of BGP lies in the stability of its guidelines with central coordination as defined by Gao & Rexford [11].

Hence, to be adopted by the NSPs, a new system must demonstrate the same abilities as BGP and even more:

- **Implementability**: It is not sufficient to solely provide models anticipating future economics and have some analytical conclusions on the model properties; NSPs expect to have a system that can run over models, this means that the computability of the solution must be considered and demonstrated by the authors.
- **Stability**: The proposed system must converge to a stable state with at least some assessed guidance policies, such as the ones defined in [11], or using existing well-known ones as the ones studied in this paper.
- **Profitability**: Replacing existing systems by new ones has a huge cost. The new system must demonstrate that it will provide short-term return over investment.

These requirements oriented our research to learning algorithms, which are, compared to optimization ones, able to fix long-term objectives and thus are adapted to anticipate future agreements. Meanwhile, the convergence properties of these algorithms have been demonstrated in restricted cases. Hence, we oriented our research to game theory modeling in order to analytically fix the stable conditions of the problem and, through experimentations, observe how learning algorithms, under various policies, can reach such conditions.

III. The SLA Negotiation game

A. Inter-NSP SLA negotiation game

Before defining the SLA negotiation as a game, we recall some definitions of game theory. A game in a normal form is a tuple $\langle n, A^1, \ldots, A^n, R^1, \ldots, R^n \rangle$ where n is the number of players, $A^i (i = 1, \ldots, n)$ is the set of available actions of the player i and $R^i : A^1 \times \cdots \times A^n \rightarrow \Re$ is the payoff function of the player i.

Game theory is a tool allowing to define the stability conditions in a competitions. Let a^i denote the action chosen by the player i, $a = (a^1, \ldots, a^n)$ denotes an action profile and $a^{-i} = (a^1, \ldots, a^{i-1}, a^{i+1}, \ldots, a^n)$ denotes the actions chosen by all the players except i. A *pure Nash equilibrium* is defined as an action profile $a_* = (a^1_*, \ldots, a^n_*)$ such that $\forall i = 1, \ldots, n, \forall a^i \in A^i, \ , R^i(a^i_*, a^{-i}_*) \geq R^i(a^i, a^{-i}_*)$.

Mixed Nash Equilibria. Let $\pi = (\pi^1, \ldots, \pi^n)$ be a strategy profile where π^i is the *strategy* of the player i, s.t. $\sum_{a^i \in A^i} \pi^i(a^i) = 1$, $\pi^i(a^i)$ being the probability of the player i to select the action a^i. A strategy profile $\pi_* = (\pi^1_*, \ldots, \pi^n_*)$ is a *mixed Nash equilibrium* if: $\forall i = 1, \ldots, n$ and $\forall \pi^i$,

$$\sum_{a^1 \in A^1} \cdots \sum_{a^n \in A^n} \pi^i_*(a^i) \pi^{-i}_*(a^{-i}) R^i(a^i, a^{-i}) \geq$$

$$\sum_{a^1 \in A^1} \cdots \sum_{a^n \in A^n} \pi^i(a^i) \pi^{-i}_*(a^{-i}) R^i(a^i, a^{-i})$$

where:

$$\pi^{-i}(a^{-i}) = \prod_{j \neq i} \pi^j(a^j)$$

SLA Negotiation as a game. Fig. 1 illustrates an SLA negotiation game: 4 NSP networks are interconnected, one NSP is the customer of a service (e.g. a pipe for gaming), another one is the target and both are linked through 2 intermediate networks. For the demanded service, each intermediate NSP proposes three equivalent SLAs defined by a QoS level and a price. The first SLA (denoted by Pa) has a weak QoS level (e.g. high delay, low bandwidth) and a low price. The second SLA (denoted by Sc) has a medium QoS level (e.g. medium delay, medium bandwidth) and a middle price. Finally, the third SLA (denoted by Ro) guarantees a high QoS (e.g. low delay, high bandwidth) but at an expensive price. The customer prefers SLA Sc (resp. SLA Ro) to SLA Pa (resp. SLA Sc) because the price difference is low compared to the QoS one. However, the customer prefers SLA Pa to SLA Ro because in this situation, it estimates that the third SLA proposes expensive services comparatively to the first one. If both NSPs propose the same SLA, then no significant gain is obtained since under several service demands customers

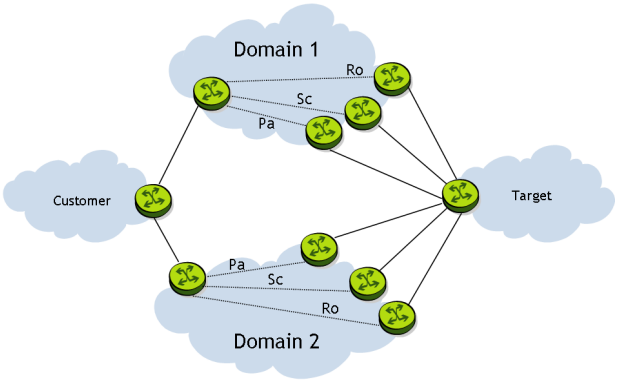


Fig. 1. SLA negotiation game

		Player 2		
		Paper	*Scissors*	*Rock*
Player 1	*Paper*	0,0	0,1	1,0
	Scissors	1,0	0,0	0,1
	Rock	0,1	1,0	0,0

TABLE I
INTER-NSP SLA PAYOFFS.

arbitrarily select one. Table I represents the utility of the intermediate NSPs.

Table I is equivalent to the well-known *Rock-Paper-Scissors' game* (RPS game) as in [12]. In this game, the player 1 picks a row and the player 2 picks a column. The intersection shows the payoff of the two players. This game has one mixed Nash equilibrium $\pi_* = (\pi_*^1, \pi_*^2)$ with $\pi_*^i(Paper) = \pi_*^i(Scissors) = \pi_*^i(rock) = \frac{1}{3}, i = 1, 2$ but has not pure Nash equilibrium[13].

Hence, as the SLA negotiation game can be modeled as a RPS game, the desirable stability conditions are probabilistic. In order to converge to such stable state, we thus opt for studying the properties of convergence of existing algorithms to such kind of state.

B. Related work

It is well-known that computing pure Nash Equilibria for 2-player zero-sum games can be reduced to a linear programing problem. To compute Nash Equilibria for 2-player general-sum games, the Lemke-Howson algorithm, inspired by the Simplex algorithm, uses pivot techniques. The authors of [14] generalized it for n-player general-sum games. Such algorithms, and exact ones in general, have an exponential complexity in the worst case. Hence, other authors, like Scarf & al.[15], designed and studied approximation algorithms to compute Nash Equilibria. However, it appears that computing an approximation of a Nash equilibrium is also exponential.

In order to reduce the computation time to find or approximate Nash Equilibria on the one hand, and distribute the computation process on the other hand, other authors focused on learning schemes. Sastry & al.[16] provided a decentralized version of the Learning Reward Inaction (LRI) algorithm and adapted it to the problem of computing mixed or pure Nash Equilibria in a game. Littman & al.[17] applied the Q-Learning algorithm for 2-player zero-sum stochastic games. Hu & al. [18] extended this work proposing an adaptation of the algorithm for 2-player general-sum stochastic games and proving its convergence to pure and mixed Nash equilibria but under specific assumptions: the Nash equilibrium is optimal (each player has his optimal payoff in the Nash equilibrium) or saddle (if a player deviates the other one improves his payoff). These results were generalized to multi-player general-sum stochastic games (i.e games whose state changes according to a probabilistic transition) by [19] but with the same assumptions.

Nanduri & al.[20] proposed a version of the Q-Learning algorithm for computing pure and mixed Nash Equilibria in matrix and stochastic games. They provided empirical results illustrating good convergence properties to pure Nash Equilibria. However, the algorithm does not seem to converge to mixed Nash Equilibria.

In the field of networking, Koutsoupias & al.[21] studied learning algorithms for the load balancing problem in networks and provides a ratio between optimal balancing and balancing obtained in Nash Equilibria. Barth & al.[22] proposed a distributed learning algorithm for inter-AS rooting games and proves its convergence to Wardrop Equilibria.

These works applied Reinforcement Learning algorithms to Game Theory. However, few of them focused on learning mixed Nash Equilibria for which the convergence of these algorithms remains an active research area.

C. Complexity of Computing Nash Equilibria

The Nash Equilibria are stable states in non-cooperative games in the meaning that each player has no interest in changing his strategy if the other players do not. A n-player game is a configuration in which each player chooses an action according to a strategy and expects a payoff. A *pure Nash equilibrium* is a configuration where there is no incentive for any player to unilaterally change its strategy: each player's action is the best response to the other players' actions. Pure Nash equilibria do not always exist. But *mixed Nash Equilibria* always exist if the game is finite.

Computing mixed Nash equilibria is difficult because it requires to explore mixed strategies in continuous domains under several constraints. All the known algorithms for computing Nash equilibria are exponential. In Algorithmic Game Theory, computing a Nash equilibrium is an important problem : *"If an equilibrium concept is not efficiently computable, much of its credibility as a prediction of the behavior of rational agents is lost"* [23]. In general, this computational problem corresponds to find a fixed point (see the Nash's proof [24]) known to be a hard problem [25]. Computing a mixed Nash equilibrium in an asymmetric 2-player game is PPAD-Complete [26] while computing a pure in a symmetric game is NP-Complete [27].

IV. LEARNING ALGORITHMS

The Reinforcement Learning algorithms are an alternative to the hardness of computing Nash equilibria. In Reinforcement

Learning, an agent interacts with his environment by performing actions and observing the environment feedback. The feedback is modeled by a reward and a new state observed by the agent. In order to improve his expected reward and learn which action to perform at each state, the agent adapts his behavior according to this feedback. The environment is thus modeled as a Markov Decision Process (MDP)[28], which is defined by a set of states, a set of available actions at each state, a reward function and a Markovian probability function to reach the next state depending upon the previous state and the previous action. This transition probability function represents the non-deterministic part of the environment, i.e., the change of state do not entirely depend upon the agent's choices. In this paper, we focus on the Q-Learning, SARSA and the LRI algorithms because they are model-free algorithms, i.e., they do not need a model of the transition function.

A. Markov Decision Processes

Formally, an MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}(.,.), \mathcal{P}(.,.,.) \rangle$ where:

- $\mathcal{S}$ is a finite set of states modeling the system states.
- $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$ is a finite set of actions representing the decisions the agent can take. We denote $\mathcal{A}_s$ the set of actions available at the state $s \in \mathcal{S}$, which is the set of possible decisions when the environment is at the state s.
- $\mathcal{R}(s,a)$, $s \in \mathcal{S}$, $a \in \mathcal{A}$ is a reward function representing the reward the agent obtains when applying the action a and being in the state s.
- $\mathcal{P}(s,a,s')$ is a conditional transition probability function to reach the state s' when performing the action a at the state s, which verifies the Markov property.

The decisions are periodically performed at "decision epochs". If the number of epochs is bounded, the MDP is a said to be a "finite horizon" MDP. Most MDP properties are preserved at an infinite horizon. For simplification purpose, we consider further in this paper finite horizon MDPs. At each epoch t, the agent is in a state denoted s_t and selects an action denoted a_t. Thus, an instantaneous reward $r_t = \mathcal{R}(s_t, a_t)$ is received by the agent and the next state s_{t+1} is reached with probability $\mathcal{P}(s_t, a_t, s_{t+1})$.

The objective of the Reinforcement Learning algorithms is to learn the policy Π^* that specifies the way the learning agent behaves by establishing a mapping between the states and the actions, and which maximizes the expected discounted sum of rewards over a finite horizon:

$$\sum_{t=0}^{T} \gamma^t \mathcal{R}^t(s_t, a_t) \cdot \mathcal{P}(s_t, a_t, s_{t+1}) \tag{1}$$

where γ denotes the discount rate (i.e. weighting future rewards) and satisfies $0 < \gamma < 1$. Formally, a *stochastic policy* Π (also called *mixed policy*) is a tuple $< \Pi^{s_1}, \ldots, \Pi^{s_{|\mathcal{S}|}} >$ where each $\Pi^{s_i} (i = 1, \ldots, |\mathcal{S}|)$ is a probability distribution over $\mathcal{A}_{s_i} \subset \mathcal{A}$. A policy Π is *deterministic* – and corresponds to a *pure* strategy in game theory terminology – if $\forall i = 1, \ldots, |\mathcal{S}|$, $\exists a_i \in \mathcal{A}_{s_i}$ $s.t.$ $\Pi^{s_i}(a_i) = 1$. i.e., Π associates always the same action to each state.

We denote:

$$v_\gamma^\Pi = E^\Pi(\sum_{t=0}^{T} \gamma^t (r_t))$$

the expected *discounted total reward* with respect to Π. We consider further that a strategy Π_* is *optimal*[1] for the discounted total reward if it maximizes v.

Puterman [29] demonstrates that if the reward function $\mathcal{R}$ is deterministic (if $s_t = s_{t'}$ and $a_t = a_{t'}$ then $r_t = r_{t'}$) then an optimal deterministic policy always exists

B. Generic algorithm

This section details the learning algorithms that we investigate: the Linear-Reward Inaction (LRI) algorithm and two Reinforcement Learning algorithms, the Q-learning and SARSA algorithms. These algorithms were chosen among others as they are "model-free", so they do not need a model of the probabilistic transition function.

Even if they differ in some formulas, the LRI and Reinforcement Learning algorithms obey to a common framework detailed by the algorithm 1. The learning scheme 1 has three main steps within a finite or infinite loop (depending on whether the MDP is at finite horizon or not).

Algorithm 1 Learning scheme

```
Initialization
loop
    at each decision epoch t
    Select an action a_t according to a policy
    Observe reward r_t and new state s_t
    Update learning data according to an update formula
end loop
```

C. Linear Reward Inaction (LRI) algorithm

The LRI algorithm learns which action maximizes the expected reward. It builds a vector of probabilities, denoted $p_t^{\mathcal{A}}$. At each decision epoch t, an action is selected according to the probability distribution of $p_t^{\mathcal{A}}$. The actualization of $p_t^{\mathcal{A}}$ is performed after the observation phase, as follows:

$$p_{t+1}(a) = p_t(a) + b \cdot r_t(1 - p_t(a)) \text{ , where } a = a_t$$

$$p_{t+1}(a') = p_t(a')(1 - b \cdot r_t) \quad \forall a' \neq a_t$$

where $0 \leq r_t \leq 1$ and $0 < b \leq 1$. The learning rate b determines the speed of learning. The LRI algorithm always converges to a distribution corresponding to a pure strategy [30](or a deterministic policy in the MDP terminology) but not always for the optimal strategy (or a pure Nash equilibrium). The probability to converge to a wrong action decreases as the parameter b decreases.

[1]An *average total reward* can also be defined. The optimal policies for average and discounted total reward are different in general. But, if the expected rewards are independent at each decision epoch, the optimal policies for the average and the discounted total reward are the same. We further consider only the discounted total reward and call *optimal policies* the optimal policies for it.

D. Reinforcement Learning algorithms: Q-Learning and SARSA

As mentioned in Sec. III-B, Reinforcement Learning algorithms have been recently studied in the context of game theory. We focus on the Q-learning and Sarsa algorithms because of their model-free ability which make them particularly adapted to the application of game theory in the inter-domain SLA negotiation.

Q-Learning algorithm. The Watkins' Q learning algorithm [31] learns the optimal Q-values of each pair (state, action) at each decision epoch t. The Q-value of a pair (state, action) at decision epoch t with respect to the policy Π is defined as

$$Q_t^{\Pi}(s,a) = E^{\Pi}[R_t | s_t = s, a_t = a]$$

where $R_t = \sum_{k=0}^{T} \gamma^k r_{t+k}$. An optimal policy Π^* is derived from the optimal Q-values, $Q^*(s,a)$:

$$\Pi^*(s) = \arg\max_{a \in \mathcal{A}_s} Q^*(s,a) \quad (2)$$

Several optimal policies Π^* can exist, but the Q-values of the actions with respect to these policies are the same.

The learning is therefore related to the Q-value function which is updated according to formula (3). This equation can be obtained using the recursive properties of the definition above. Thus, a Q-value indicates the potential gain of choosing an action when being in a given state.

$$Q_{t+1}(s,a) = (1-\alpha_t) Q_t(s,a) + \alpha_t (r_t + \gamma \max_{a' \in \mathcal{A}_{s'}} Q_t(s',a')) \quad (3)$$

Hence, the Q-Learning algorithm builds a "Q-table" for any state-action pair and maintains this table according to 3 using the observed reward r_t and next state s', a discount factor γ and a "learning-rate" denoted α. This latter also evolves at each decision epoch. As discussed below, the way α is actualized particularly impacts the convergence properties of the algorithm.

The actions are chosen at each decision epoch following a policy based on the Q-values. Various kinds of "Q-based" policies permit to explore the state/action space in order to learn the optimal policy. They are further described in Sec. IV-E. In fact, Equation 2 allows the system to reach a deterministic optimal policy. But if the considered MDP is a model of a game that has a mixed Nash equilibrium, there is necessarily another optimal policy, which is mixed. This policy corresponds to the mixed Nash equilibrium strategy of the agent.

Convergence. The Q-Learning algorithm is proven to converge to optimal Q-values under two assumptions, as demonstrated in [32]:

- All the pairs (state, action) must be visited infinitely,
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

This suggests that, in a finite-horizon MDP, the algorithm has been "trained" prior to its execution. The convergence proof of the Q-learning algorithm has been refined by the authors of [33], who demonstrated the existence of an upper bound according to how the learning rate α is updated. If the learning rate α is polynomial ($\alpha_t = \frac{1}{(t+1)^{\omega}}$, with $\frac{1}{2} < \omega < 1$) then the convergence time is polynomial in $\frac{1}{1-\gamma}$. If $\omega = 1$ then the convergence time is exponential in $\frac{1}{1-\gamma}$. If it converges to the optimal Q-values, and if all others players play their mixed Nash equilibrium strategies, then the problem is to know how using this values to find an optimal policy. This problem is closely related to the choice of a Q-based policy for which different schemes have been proposed as discussed in Sec. IV-E.

SARSA algorithm. The SARSA (State-Action-Reward-State-Action) algorithm also uses the state-action (Q) function described above. It differs from the Q-Learning algorithm in the update of the Q-values, and more specifically in the consideration of the future action. The Q-values are updated as follows:

$$Q_{t+1}(s,a) = (1-\alpha_t) Q_t(s,a) + \alpha_t (r_t + \gamma Q_t(s',a'))$$

where $a = a_t$ and $a' = a_{t+1}$. Hence, the value used for the actualization is not the one of the action maximizing the future reward but the one of the "real" next action. This suggests some smooth modifications in Algorithm 1 to keep in memory the past action and reward when the update is performed.

The authors of [34] provided a proof of convergence of the SARSA algorithm under the following assumptions:

- All pairs (state, action) must be visited infinitely,
- The Q-based policy to choose the actions is greedy at the limit (when $t \to +\infty$).

E. Q-based policies

A Q-based policy is the way to select an action based on the Q-values. This policy must allow exploration of the environment (trying several actions to learn Q-values) but also exploitation (choosing frequently actions that maximize the expected reward). Initially, the agent has to learn, so the exploitation has priority. After convergence, the Q-values are precise enough to be exploited, so exploitation has priority. We focus on the four must used Q-based policies and discuss their capacity to reach a mixed optimal policy.

Greedy policy. The greedy policy selects always the action having the highest Q-value, such as $a_t = \underset{a \in \mathcal{A}_s}{\operatorname{argmax}}\ Q_t(s,a)$.

ϵ-greedy policy. The ϵ-greedy policy selects the action having the highest Q-value with probability $1-\epsilon$, and a random action with probability ϵ. ϵ is initialized to a high value in order to encourage exploration. It decreases as the Q-values become more precise. The policy is greedy at the limit.

Softmax policy. The softmax policy, defined as $P(a_t = a) = \frac{Q_t(s,a)}{\sum_{a' \in \mathcal{A}_s} Q_t(s,a')}$, selects an action with proportional probability to its Q-value. This policy is not necessarily greedy at the limit.

Boltzmann policy. The Boltzmann policy selects an action using the formula $P(a_t = a) = \frac{e^{Q_t(s,a)/\tau}}{\sum_{a' \in \mathcal{A}_s} e^{Q_t(s,a')/\tau}}$, where $\tau \in \Re_+^*$ governs the lag between action probabilities. When $\tau \to \infty$, the lags between the probabilities tend to 0 and the policy is nearly random and uniform. When $\tau \to 0$, the lags between the probabilities increase and the policy is nearly

greedy. Usually, τ is initialized to a high value to encourage exploration, then it is decreased to tend to a greedy policy.

Policies and convergence. The greedy policy may not satisfy the convergence requirements (each action must be selected infinitely) previously mentioned. In addition, if the algorithm converges to the optimal Q-values then the action selection will be deterministic, which corresponds to a pure strategy (unless there are several actions that share the same highest Q-value).

If the chosen Q-based policy is ϵ-greedy, then the SARSA Algorithm convergence requires that $\epsilon \to 0$ when $t \to +\infty$, which gives a deterministic policy and therefore a pure strategy. The convergence of the Q-Learning algorithm does not require that. But the action with the highest Q-value is selected with probability $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}_s|}$. The other actions are selected with probability $\frac{\epsilon}{|\mathcal{A}_s|}$. These probabilities do not necessary correspond to a mixed Nash Equilibrium.

A softmax policy allows to have a mixed strategy if there is more than one action that has a nonzero Q-value. However, if the opponent players play their mixed Nash equilibrium strategies, $E(r_t|a_t = a)$ are the same $\forall a \ s.t. \ \pi^i_*(a) \neq 0$ (see the Nash's proof for the support theorem [24]). Thus, their Q-values are also the same. If the learning algorithm converges to the correct Q-values, then softmax policy leads to a uniform probability distribution strategy over $supp(\pi^i_*) = \{a \mid \pi^i_*(a) \neq 0\}$, which is not necessarily a mixed Nash equilibrium strategy.

The same problem arises if Boltzmann policy is used. If the Q-values are strictly the same then Boltzmann policy leads to a uniform random strategy. However, the convergence after a finite computation time may be not complete. The Q-values may be close but not strictly identical. With Boltzmann policy, even if $Q^*(s, a)$ is very close to $Q^*(s, a')$, the difference between $p(a)$ and $p(a')$ can be high. If $\tau \to +\infty$, this policy becomes greedy at the limit. A good value of τ can provide a mixed Nash equilibrium strategy. But this value depends on the differences between the Q-values of the actions, and these differences (when the convergence is not complete) strongly depend on the initialization and are also random (due to the randomness of the used policy). In [35], the authors demonstrated that under symmetric conditions of τ, the Boltzmann policy converges to a sub-set of mixed Nash Equilibria. However, the quality of these equilibria remains to be considered.

V. Experimental Observations

This section relates the results of the simulation we conduced on the RPS game modeled by Table I. A RPS game is run several times (called "rounds") - ca. 10000 are illustrated in most figures on the X axis, and the number of simulations performed for each round was set at 100.

An opponent player, able to play pure or mixed strategies, plays against the observed player, which uses one of the mentioned algorithms (either LRI, Q-learning or SARSA) with various policies. Our goal is to experimentally evaluate the performances of these algorithms, i.e. their ability to converge, to learn against a player implementing specific strategies (e.g. a pure or a mixed strategy), including when the player applies also a learning scheme. Our study complements the one of [36] where the authors presented results opposing two players using an adapted Q-Learning algorithm (called individual Q-learning) with player-dependent learning rates (PDLR). In the specific case of a Boltzmann policy and particular games, the authors demonstrated the convergence to Nash distributions (i.e. approximation of mixed Nash equilibria).

Each learning algorithm has been evaluated using different policies:

- LRI algorithm: two versions was considered, one with a static learning rate, titled "LRI" on the figures, another with a decreasing learning rate, titled "LRI b decreasing" on the figures.
- Q-Learning algorithm: the Q-Learning algorithm were applied with an ϵ-greedy, a Softmax and a Boltzmann policies. For the ϵ-greedy policy, the ϵ value was set to 0.01 and decreased until stabilization during the first half of the experiment duration. For the Boltzmann policy, the τ value was set to 0.9. The learning rate was initialized at 0.6 and updated as recommended by [33]. Different experiments was conduced with different discount factor values.
- SARSA algorithm: the SARSA algorithm was applied with the same policies and parameters than the Q-Learning algorithm.

A. Reward convergence

Fig. 2 shows the gains obtained by the observed player when using the different learning algorithms facing different Opponent player strategies. On the Y axis of these figures is represented the reward obtained each 100 round, which is accordingly to Table I 100 in the best case. Fig. 2 exhibits that most algorithms bring more gain than the mixed Nash equilibrium, except the Q-learning algorithm with a Boltzmann policy when the opponent player plays the mixed Nash equilibrium or a pure strategy, and the SARSA algorithm with a Boltzmann policy when the opponent player plays a mixed strategy. However, as illustrated by Fig. 2(a), 2(b) and 2(c), the Q-Learning algorithm with a Boltzmann policy is the Reinforcement Learning algorithm providing the highest gains when the opponent player plays the other strategies.

All the results also validate that the LRI algorithm is the one bringing the highest gains whatever the strategy of the opponent player is, and that the policy used in the Reinforcement Learning algorithm affects a lot the obtained results. Note also, that using a decreasing learning rate with the LRI algorithm does not significantly improve or damage the obtained gains.

Finally, Fig. 2(b) exposes the results obtained when the Reinforcement Learning algorithms use a discount factor set at 0. This makes their behavior closer to the LRI algorithm where only instantaneous gain is optimized. It is also better adapted to the RPS game where future gains are totally independent from past the ones. Fig. 2(b) shows that when the discount factor is set at 0, most of the Reinforcement Learning

algorithms present higher gain results but are still less efficient than the LRI algorithm.

B. Learning Strategy

In order to evaluate the ability of the algorithms to learn facing up various kind of strategies, we conduced experiments where the Opponent player applies a pure strategy, the "Rock" strategy, and when he plays his mixed Nash equilibrium strategy. Fig. 3 illustrates the simulation results when the opponent player plays the "Rock" strategy.

If the opponent player applies the pure strategy, we observed how the learning schemes learn that "scissors" is the strategy to avoid (cf. Fig. 3(a)) and "paper" the strategy to win (cf. Fig. 3(b)). Fig. 3(b) exhibits that the LRI algorithms learn rapidly to apply the "paper" strategy while the Reinforcement Learning algorithms converge slower. The softmax policy seems to allow a faster convergence than the other Reinforcement Learning policies, especially than the Boltzmann policy, which almost stabilizes. The ϵ-policy provides in-between results. Both softmax and ϵ-greedy policies are still learning at 10000 rounds. Hence, tuning the parameters of these policies could speed up the convergence. We let such simulation studies for future work.

We conduced similar experiments when the opponent player applies the mixed Nash equilibrium. However, the results were not significant enough to be presented here: most algorithms converge to a - different at each simulation - pure strategy except the LRI with a decreasing factor, which converges to a mixed strategy (but not the mixed Nash equilibrium one).

C. Opponent player applies a learning scheme

In order to complement this study, we evaluated the learning algorithms when the opponent player uses also a learning algorithm. Fig. 4 illustrates these results. On the Y-axis of these figures is represented the reward obtained each 100 round, which is of 100 in the best case according to table I.

On Fig.4(a), one might observes that the algorithms obtaining the highest reward against the LRI are the Q-Learning and the SARSA algorithms associated to ϵ-greedy policy. Fig. 4(b) shows that the Q-learning algorithm associated to Boltzmann policy obtains the highest rewards.

Figures 4(c) and 4(d) show the simulation results when the opponent player uses the Q-Learning algorithm with different policies. The algorithms obtaining the highest reward against ϵ-greedy policy are the Q-learning with the same policy and the SARSA algorithm with Boltzmann policy. The latter also performs the best against the softmax policy. And the algorithm obtaining the highest reward against the Boltzmann policy is the Q-Learning with Boltzmann policy. Only the LRI opposed to a softmax policy is represented on Fig. 4(c) since, against other policies, the LRI does not bring any reward. The LRI with a decreasing learning rate performs a bit better but still only against a softmax policy and not so significantly.

Figures 4(e) and 4(f) exhibit results when the opponent player uses the SARSA algorithm with various policies. The algorithm performing the best against the SARSA algorithm with ϵ-greedy policy is the Q-Learning with Boltzmann policy. It is also the one getting the highest gains against the softmax policy, together with the SARSA algorithm with Boltzmann policy. As for the comparison with the Q-Learning, only partial results of the LRI algorithm are represented on Fig. 4(e). In the case of an opponent player using the SARSA algorithm, the LRI algorithms perform a bit better against both softmax and ϵ-greedy policies.

Hence, Boltzmann policy can be considered as the most suitable for Reinforcement Learning algorithms when the opponent uses also a Reinforcement Learning algorithm. When the opponent uses the LRI algorithm, ϵ-greedy policy should be preferred.

VI. Conclusion

In the context of inter-NSP SLA negotiation, ensuring fair revenues and stability is the key challenge for the adoption of new business models. Reaching Nash Equilibria is the best way to ensure this stability but pure Nash equilibria do not always exist and some equilibria are more profitable than others. Recently, several works focused on Reinforcement Learning algorithms to solve the Inter-NSP SLA negotiation problem. In this paper, we investigated the capacity of Reinforcement Learning algorithms to learn mixed Nash equilibria. We experimented model-free learning algorithms to solve the Rock-Paper-Scissors' game, to which an inter-NSP SLA Negotiation can be reduced. This game is known to have a mixed Nash equilibrium. We observed that all these algorithms fail to converge to the unique mixed Nash equilibrium of the game even though they provide high gains to the NSPs. We also observed that the Boltzman policy used with the Q-learning solutions seems to be the most stable and profitable solution in this modeling of the SLA negotiation problem.

In future work, we aim to investigate the theoretical assessment of the observations detailed in this paper. In particular, we aim to focus on the Boltzman policy, which according to our experiments seems the most suitable for this problem. As we developed in the motivation of this paper, this theoretical assessment is crucial to demonstrate the relevance of the approach. We will also study how the learning algorithms behave in the case of the existence of equilibria having different profitability properties.

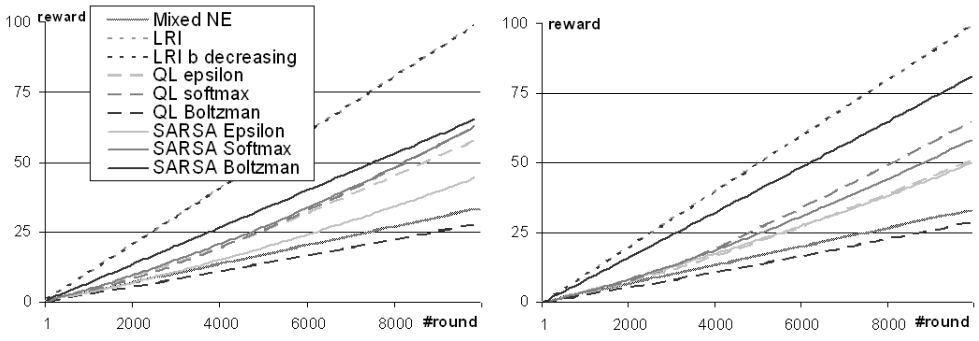
VII. Acknowledgments

We are grateful to Johanne Cohen for her feedback on earlier versions of this paper. This work has been partially supported by the ETICS-project (Grant agreement no.: FP7-248567, Contract Number: INFSO-ICT-248567), granted by the European Commission.

References

[1] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet inter-domain traffic. In *SIGCOMM*, pages 75–86, 2010.

[2] D. Barth, L. Echabbi, and C. Hamlaoui. Optimal transit price negotiation: The distributed learning perspective. *Journal of Universal Computer Science*, 14:745–765, 2008.



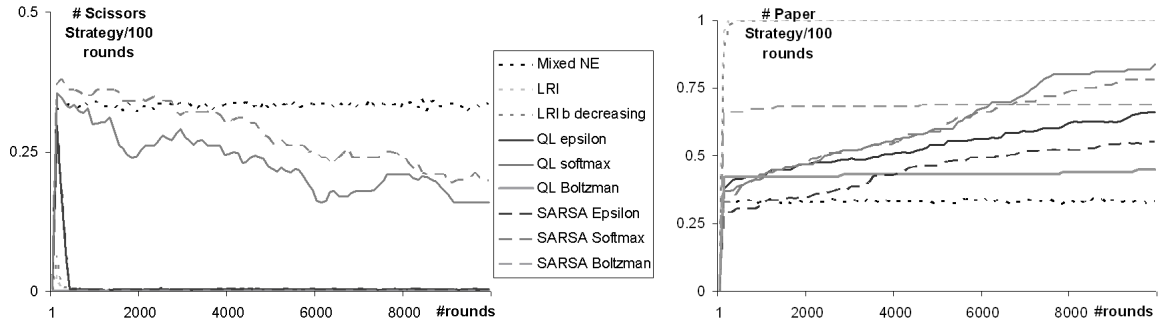
(a) Opponent plays mixed NE strategy

(b) Opponent plays mixed NE strategy, $\gamma = 0$

(c) Opponent plays pure strategy

(d) Opponent plays mixed strategy

Fig. 2. Simulation results: obtained gains



(a) Agent plays scissor

(b) Agent plays paper

Fig. 3. Opponent player applies a pure "Rock" strategy

[3] R. Ma, D. Chiu, J. Lui, V. Misra, and D. Rubenstein. Internet Economics: The use of Shapley value for ISP settlement. In *CoNEXT*, 2007.

[4] Vytautas Valancius, Nick Feamster, Ramesh Johari, and Vijay V. Vazirani. MINT: a Market for INternet Transit. In *CoNEXT*, page 70, 2008.

[5] H. Pouyllau and R. Douville. End-to-end QoS negotiation in network federations. In *IEEE NOMS Bandwidth on Demand (BoD) Workshop*, 2010.

[6] Zied Ben Houidi and Helia Pouyllau. The price of tussles: bankrupt in cyberspace ? In *ACM SIGMETRICS/Performance Workshop on Pricing and Incentives in Networks*, 2012.

[7] Andrew Odlyzko. Internet pricing and the history of communications, 2001.

[8] I.Ch. Paschalidis and J.N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking*, 8(2):171 – 184, apr 2000.

[9] S. Kunniyur and R. Srikant. End-to-end congestion control schemes: utility functions, random losses and ecn marks. *Networking, IEEE/ACM Transactions on*, 11(5):689 – 702, oct. 2003.

[10] Zhi-Li Zhang, Papak Nabipay, Andrew M. Odlyzko, and Roch Guérin. Interactions, competition and innovation in a service-oriented internet: An economic model. In *INFOCOM*, pages 46–50, 2010.

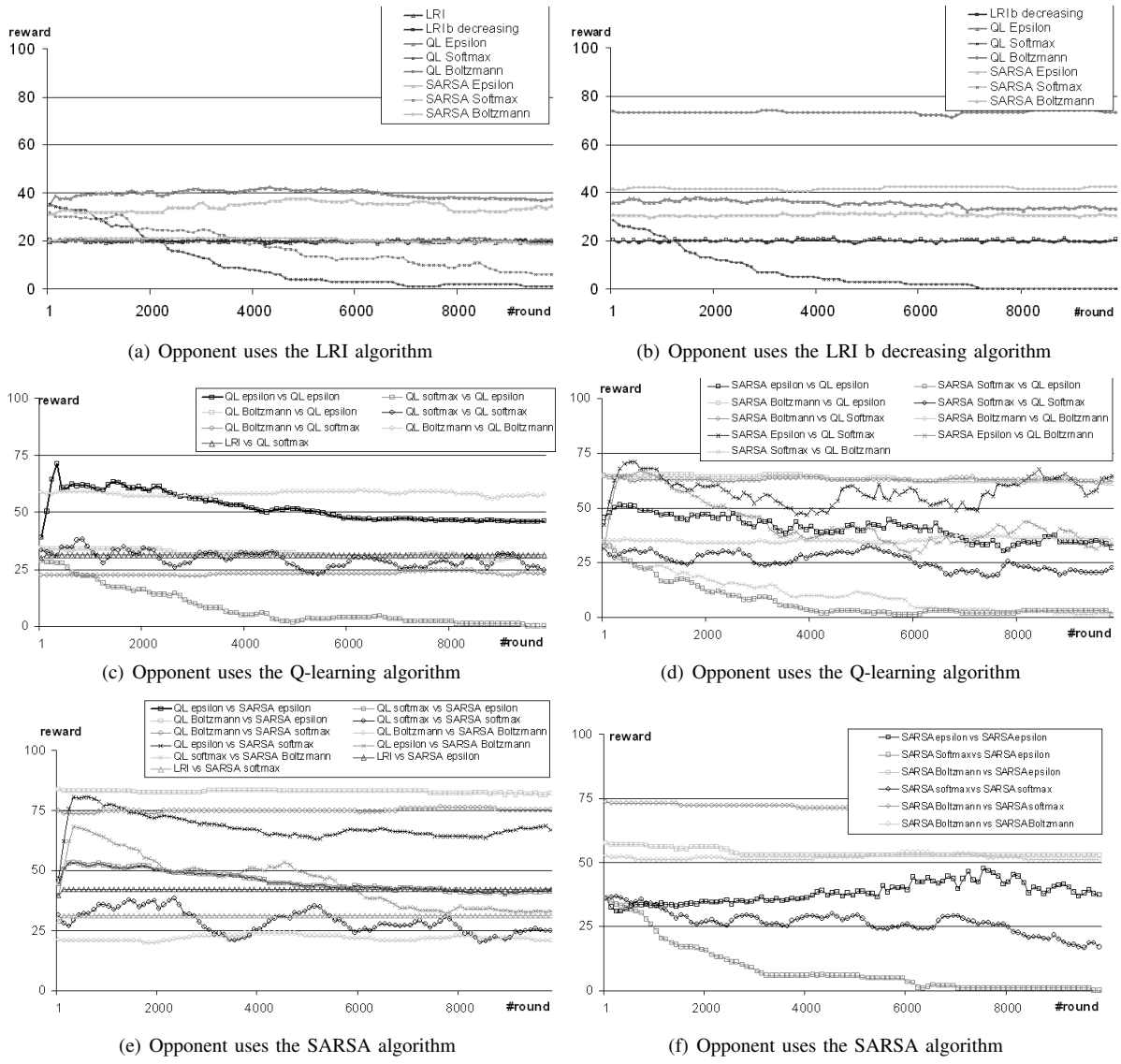
[11] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. In *IEEE/ACM Transactions on Networking*, pages 681–692, 2000.

[12] B. Codenotti and D. Stefankovic. On the computational complexity of Nash equilibria for (0, 1) bimatrix games. *Inf. Process. Lett.*, 94(3):145–150, 2005.

[13] D.C. Fisher and J. Ryan. Optimal Strategies for a Generalized "Scissors, Paper, and Stone" Game. *The American Mathematical Monthly*, 99(10):pp. 935–942, 1992.

[14] J. Rosenmuller. On a Generalization of the Lemke-Howson Algorithm to Non-cooperative N-Person Games. *SIAM Journal on Applied Mathematics*, 21(1):pp. 73–79, 1971.

[15] H.E. Scarf and T. Hansen. *The Computation of Economic Equilibria*.



(a) Opponent uses the LRI algorithm

(b) Opponent uses the LRI b decreasing algorithm

(c) Opponent uses the Q-learning algorithm

(d) Opponent uses the Q-learning algorithm

(e) Opponent uses the SARSA algorithm

(f) Opponent uses the SARSA algorithm

Fig. 4. Reward obtained when the Opponent player uses a learning algorithm

Yale University Press, 1973.

[16] P. S. Sastry, V. V. Phansalkar, and M. A. L. Thathachar. Decentralized Learning of Nash Equilibria in Multi-Person stochastic games with incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics*, 24:769–777, 1994.

[17] M. L. Littman and C. Szepesvári. A Generalized Reinforcement-Learning Model: Convergence and Applications. In *ICML*, pages 310–318, 1996.

[18] J. Hu and M. P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *ICML*, pages 242–250, 1998.

[19] J. Hu and M. P. Wellman. Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[20] V. Nanduri and T.K. Das. A Reinforcement Learning algorithm for obtaining the Nash Equilibrium of multi-player matrix games. *IIE Transactions*, 41(2):158–167, 2009.

[21] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *STACS*, pages 404–413, 1999.

[22] D. Barth, J. Cohen, O. Bournez, and O. Boussaton. Distributed Learning of Equilibria in a Routing Game. *Parallel Processing Letters*, 19(2):189–204, 2009.

[23] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[24] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[25] Michael D. Hirsch, Christos H. Papadimitriou, and Stephen A. Vavasis. Exponential lower bounds for finding Brouwer fixed points. *J. Complexity*, 5(4):379–416, 1989.

[26] X. Chen and X. Deng. Settling the Complexity of 2-Player Nash-Equilibrium. *Electronic Colloquium on Computational Complexity*, 140, 2005.

[27] I. Gilboa and E. Zemel. Nash and Correlated Equilibria: Some Complexity Results,. *Games and Economic Behavior*, 1989.

[28] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[29] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[30] L. Pack Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement Learning: A Survey. *J. Artif. Intell. Res. (JAIR)*, 4:237–285, 1996.

[31] C. J.C.H. Watkins and P. Dayan. Technical Note: Q-Learning. In *Journal of Machine Learning Research*, pages 279–292, 1992.

[32] M. L. Littman. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *ICML*, pages 157–163, 1994.

[33] E. Even-Dar and Y. Mansour. Learning Rates for Q-learning. *Journal of Machine Learning Research*, 5:1–25, 2003.

[34] S. P. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. *Machine Learning*, 38(3):287–308, 2000.

[35] Roberto Cominetti, Emerson Melo, and Sylvain Sorin. A payoff-based learning procedure and its application to traffic games. *Games and Economic Behavior*, 70(1):71 – 83, 2010. ¡ce:title¿Special Issue In Honor of Ehud Kalai¡/ce:title¿.

[36] D. S. Leslie and E. J. Collins. Individual Q-learning in normal form games. *Journal on Optimal Control*, 44:459–514, 2004.