Ca' Foscari
University
of Venice

Department
of Economics

Working Paper

Marco Corazza
Gianni Fasano
Riccardo Gusso
Raffaele Pesenti

A comparison among Reinforcement Learning algorithms in financial trading systems

ISSN: 1827-3580
No. 33/WP/2019

Working Papers
Department of Economics
Ca' Foscari University of Venice
No. 33/WP/2019
ISSN 1827-3580

A comparison among Reinforcement Learning algorithms in financial trading systems

Marco Corazza
Department of Economics
Ca' Foscari University of Venice

Gianni Fasano
Department of Management
Ca' Foscari University of Venice

Riccardo Gusso
Department of Economics
Ca' Foscari University of Venice

Raffaele Pesenti
Department of Management
Ca' Foscari University of Venice

First Draft: November 2019

Abstract
In this work we analyze and implement different Reinforcement Learning (RL) algorithms in financial trading system applications. RL-based algorithms applied to financial systems aim to find an optimal policy, that is an optimal mapping between the variables describing the state of the system and the actions available to an agent, by interacting with the system itself in order to maximize a cumulative return. In this contribution we compare the results obtained considering different on-policy (SARSA) and off-policy (Q-Learning, Greedy-GQ) RL algorithms applied to daily trading in the Italian stock market. We consider both computational issues related to the implementation of the algorithms, and issues originating from practical application to real stock markets, in an effort to improve previous results while keeping a simple and understandable structure of the used models.

Keywords
Reinforcement Learning, SARSA, Q-Learning, Greedy-GQ, financial trading system, Italian FTSE Mib stock market

JEL Codes
C53, C54, E37, G17

Address for correspondence:
Riccardo Gusso
Department of Economics
Ca' Foscari University of Venice
Cannaregio 873, Fondamenta S.Giobbe
30121 Venezia - Italy
Phone: (++39) 041 2349199
Fax: (++39) 041 2349176
E-mail: rgusso@unive.it

The Working Paper Series
is available only on line
(www.unive.it/pag/16882/?L=1)
For editorial correspondence, please contact:
wp.dse@unive.it

Department of Economics
Ca' Foscari University of Venice
Cannaregio 873, Fondamenta San Giobbe
30121 Venice Italy
Fax: ++39 041 2349210

1 Introduction

Starting from the Mid-Eighties of the past century, there has been an increasingly growing employment of intelligent approaches for creating and developing innovative methods and tools for financial problem-solving and decision-making. In particular, a large number of such methods and tools has been devoted to the creation and development of systems for automated capital management (for a survey of intelligent applications in finance since the initial phase of their appearance, see [1]).

In this paper, we propose and apply some novel automated Financial Trading Systems (FTSs) based on a self-adaptive machine learning approach generally known as Reinforcement Learning (RL). In particular, we deeply investigate potentialities and the effectiveness of automated FTSs based on the intelligent approaches *State-Action-Reward-State-Action* (SARSA) (see for instance [19, 3]) and *Q-Learning* (QL) (see for instance [25, 3]), together with its recent development *Greedy-GQ* (see [17]), which belong to the family of the RL methodologies.

The above algorithms concern an agent dynamically interacting with an environment. During this interaction, the agent perceives the state of the environment and takes an action. The environment, on the basis of such an action, provides a negative or a positive reward. This process allows the agent to heuristically detect a policy that maximizes a cumulative reward over time (for details see Section 2). In our case, the agent is a FTS and the environment is a financial market. The FTS, once perceived the state of the market, decides to sell/buy an asset or to stay out of the market. In response, the market provides a loss or a gain to the FTS. This permits the online detection of a trading strategy for the maximization over time of a cumulative performance measure (for details see Section 3).

Prior to begin the above investigation, the following crucial matter has to be dealt with: why developing trading strategies based on intelligent techniques and, in particular, on RL methodologies? The answer needs some preliminary considerations concerning the reference theoretical frameworks through which one formalizes the functioning of the financial markets.

According to the mainstream financial theory of the Efficient Market Hypothesis (EMH) in its weak form, it is not possible to systematically make profitable trading in financial markets. In fact, according to this theory, the economic agents acting in such markets are fully rational, that is, through the law of the demand and the supply, they are able to instantaneously and appropriately vary the prices of the financial assets simply on the basis of the recent information. In this theoretical framework, the only source of (unpredictable) variations of the prices of the financial assets, between two consecutive time instants, is

represented by the arrival of unexpected new information (for more details see for instance [9]).

However, as common sense suggests and as behavioural finance substantiates, human beings – and therefore economic agents – are often non fully rational when making decisions, especially under uncertainty. In fact, since the Seventies of the past century, experimental economists have documented several departures of the real investors' behaviours from those prescribed by the EMH. The main implication coming from these departures from the EMH consists in the fact that financial markets are not so rarely inefficient, and consequently that they, more or less frequently, offer possibilities of profitable trading.

Of course, as a matter of fact, sometimes the financial markets can be inefficient, it is of some importance to give an answer to the question: how taking advantage of these possibilities of trading? The answer depends on the chosen reference theoretical framework. In our opinion, the currently most convincing attempt to reconcile the EMH with the empirical departures from it is given by the so-called Adaptive Market Hypothesis (AMH) (see for more details [13], [14], [15] and [16]). Following this novel theory, a financial market can be viewed as an evolutionary environment in which different partly rational but anyway intelligent "species" (for instance, hedge funds, market makers, pension funds, retail investors, speculators and so on) interact among them in order to achieve a satisfactory, not necessarily optimal, level of some measure of profitability. Note that, as these species are partly rational, their adaptations to the various stimuli is neither instantaneous nor immediately appropriate, and this generally does not imply the efficiency of the financial market. Because of that, the AMH entails that $\ll$[f]*rom an evolutionary perspective, the very existence of active liquid financial markets implies that profit opportunities must be present. As they are exploited, they disappear. But new opportunities are also constantly being created as certain species die out, as others are born, and as institutions and business conditions change*$\gg$ [13, p. 24]. So, coming back to the previous question, it seems reasonable that an effective FTS has to be a new intelligent specie able to real-time interact with the considered financial market, in order to learn its unknown and structurally time-varying dynamics, and able to exploit this knowledge in order to real-time detecting profitable financial trading policies.

Therefore, it is on the basis of the reference theoretical framework we have chosen (i.e. the AMH), and of the features of the FTS we have required, that in this paper we resort to the self-adaptive machine learning methodology-type approach known as RL for developing automated intelligent FTSs. This approach is also known as Neuro-Dynamic Programming (see for instance [5]) and Dynamic Programming Stochastic Approximation (see for instance [11]). More specifically, we consider the RL methodology to create and develop automated

FTSs as it can be viewed as a stochastic optimal control problem solver that has to discover the optimal financial trading strategies.

Note that QL, Greedy-GQ and SARSA methodologies do not provide optimal solutions but good near-optimal ones. So, last matter to tackle: why not resorting to more classical stochastic dynamic programming methods guaranteeing the achievement of optimal solutions? Generally speaking, the latter typology of methods needs the precise description of the probabilistic features of the investigated financial markets. But, as we state below, once chosen AMH as reference theoretical framework, the dynamics of financial markets are unknown and structurally time-varying, from which the impossibility of providing such a required precise description. Differently, $\ll$*RL does not need apriori knowledge of the transition probability matrices*$\gg$ [11, p. 199].

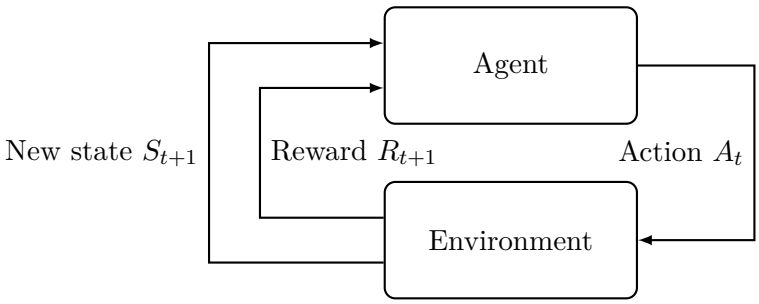
The remainder of the paper is organized as follows. In the next section, we describe the background of RL theory which QL, Greedy-GQ and SARSA are based on. In Section 3 we present our implementations of the FTSs based on the aforementioned algorithms and consider the problem of the description of the financial environment state and of its implications in terms of differences of the considered learning approaches. In Section 4 we analyze the results obtained by applying the developed FTSs to several stocks of the Italian FTSE Mib market. Finally, in Section 5 we give some final considerations.

2 Reinforcement learning background

Reinforcement Learning (see [3]) can be described as a class of machine learning solution methods developed to find optimal actions in a subset of possible choices for the current state of the system. The state is determined interacting over time with a given framework, in order to maximize a reward which can depend on future states of the framework itself. Actually, RL considers problems where the following elements can be identified: (i) the *agent*, which is a learning decision maker, (ii) the *environment* the agent interacts with in subsequent time steps, (iii) a set of possible *actions* to choose among at each time step, (iv) a feedback signal, namely the *reward*, that the environment awards responding to actions chosen by the agent. To formally describe the problems that RL methods aim to solve, the theory of finite Markov Decision Processes (MDPs) (see [18]) needs first to be considered.

The interaction between the agent and the environment occurs at finite time steps $t = 0, 1, 2, \ldots$, and at each time step t the agent is given a description of current environment *state* $S_t \in \mathcal{S}$, being $\mathcal{S}$ the set of all possible states. The agent then selects an *action* $A_t \in \mathcal{A}(S_t) \subset \mathcal{A}$ among the possible ones (i.e. $\mathcal{A}(S_t)$) in the current state, so that at the subsequent time step $t+1$ it receives both a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and the new description

of environment state S_{t+1}.



The next assumption holds within the current paper.

Assumption 2.1 *The sets $\mathcal{S}, \mathcal{A}$ and $\mathcal{R}$ have a finite number of elements. Then, random variables R_t, S_t have a discrete probability distribution conditioned only on preceding state and action, i.e.*

$$p(s', r|s, a) \coloneqq \mathbb{P}\left[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\right], \tag{1}$$

which expresses the Markov property for the states in a MDP.

A key feature of RL methods is the *reward hypothesis*: that is, the agent's objective is simply to maximize some cumulative amount of the rewards it receives over time. The most common formalization of this hypothesis is obtained by the introduction of the *discounted return*

$$G_t \coloneqq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2}$$

where $0 \leq \gamma \leq 1$ is the discount rate.

Indeed, to select how relevant is for the agent a specific action in a given state, the majority of RL algorithms computes *action-value* functions based on expected discounted returns. Since future rewards are also determined on the base of future actions, action-value functions are defined in terms of rules (namely *policies*), that assign at every time step the probability of choosing an action given a state. That is, given s as in (1), a policy $\pi(a|s)$ is a discrete probability distribution over $\mathcal{A}(s)$ for every $s \in \mathcal{S}$, and the corresponding action-value function $q_\pi(s, a)$ is defined as (see (2))

$$q_\pi(s,a) \coloneqq \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] \tag{3}$$

where the expected value $\mathbb{E}_\pi$ is meant to be computed given that the agent follows the policy π after choosing $a \in \mathcal{A}(s)$.

Using the fact (see (2)) that $G_t = R_{t+1} + \gamma G_{t+1}$, it is easy to prove (see also [3]) that action-value functions satisfy the following recursive relation between the value of the current state-action pair (i.e. (s,a)) and the possible subsequent pair (s',a'), i.e.

$$q_\pi(s,a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) \left[r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_\pi(s', a') \right]. \tag{4}$$

Equation (4) is the so called *Bellman equation* for $q_\pi(s,a)$, and for finite MDPs has a unique solution (see also [4]).

The goal of a RL algorithm is to find an *optimal* policy π', considering the partial order induced on policies by defining $\pi' \succeq \pi$ if and only if $v_{\pi'}(s) \geq v_\pi(s)$, for every $s \in \mathcal{S}$, where

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) q_\pi(s,a).$$

All optimal policies have the same optimal action-value function, which is given by $q_*(s,a) = \max_\pi q_\pi(s,a)$. Such a function satisfies the following special form of Bellman equation (see also [4])

$$q_*(s,a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) \left[r + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a') \right], \tag{5}$$

being $v_*(s) = \max_\pi \sum_{a \in \mathcal{A}(s)} \pi(a|s) q_\pi(s,a) = \max_{a \in \mathcal{A}(s)} q_*(s,a)$. Equation (5) yields actually a system of equations of cardinality $|\mathcal{S}||\mathcal{A}|$ that can in principle be solved if the dynamic conditioned probability $p(s', r|s, a)$ of the MDP is known. If $q_*(s,a)$ is given, then the optimal policy would be the *greedy* one with respect to the optimal action-value function, that is the one such that, for each state $s \in \mathcal{S}$, an action a with $a \in \arg\max_{a \in \mathcal{A}(s)} q_*(s,a)$ is selected.

However, the explicit determination of the optimal action-value function is often of difficult computation. Indeed, even if the dynamics of the MDP is known and the Markov property holds, then $|\mathcal{S}||\mathcal{A}|$ could be so large to discourage the direct computation of $q_*(s,a)$. RL methods tackle this problem by approximately determining (sub-)optimal action-value functions, using information the agent obtains by direct interaction with the environment, without assuming a complete knowledge of a model for the probability distribution of rewards and states of the system.

In order to better understand how RL algorithm works, it is useful to recall some ideas from Dynamic Programming (see [6]). Assuming that we have a finite MDP (i.e. Assumption 2.1 holds) and its dynamics $p(s', r|s, a)$ is known, suppose we need to evaluate the action-value function corresponding to a given policy π. Then, it is possible to prove (see [12]) that starting from an arbitrary approximation $q_0(s, a)$, the iterative sequence $\{q_k(s, a)\}_{k=0,1,\dots}$ can be computed such that

$$q_{k+1}(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_k(s', a') \right], \tag{6}$$

and under suitable assumptions $\{q_k(s, a)\}$ will converge to a unique fixed point, which is $q_\pi(s, a)$ as by Bellman equation (4). Once $q_\pi(s, a)$ has been determined, we can improve current policy π by replacing it with $\pi'(s, a)$, which is the *greedy* one with respect to $q_\pi(s, a)$; then, we evaluate again its action-value function, and so on. This procedure is called *policy iteration*, and under suitable assumptions it will converge to an optimal policy and to the corresponding action-value function. Indeed, it is possible to prove that the described policy evaluation and the resulting improvement steps can be combined in the following unique iterative procedure involving action-value functions

$$q_{k+1}(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma \max_{a' \in \mathcal{A}(s')} q_k(s', a') \right]. \tag{7}$$

The iteration (7) is expected to converge to the optimal action-value function, from which we can obtain the optimal greedy policy.

RL methods try to extend this approach to more general problems, where a probabilistic description of the environment is unknown and its dynamics is not given. Nevertheless, RL replaces it using the experience obtained from sample sequences of actual or simulated states, actions, and rewards $S_0, A_0, R_1, A_1, R_2, \dots$ In this contribution we will focus our attention on some methods belonging to RL Temporal-Difference (TD) family (see [22]), which can be seen as a combination of Monte Carlo RL methods (see [21]) and Decision Processes.

Since by (3) action-value functions represent the expected value of discounted returns, we may consider their estimates $Q(s, a)$ at each time step t. Then, $Q(s, a)$ can be updated using experienced or simulated rewards, in order to average them over a finite time horizon (see [3]), i.e.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha_t \left[G_t - Q(S_t, A_t) \right], \tag{8}$$

where $Q(S_t, A_t)$ is the current estimate of $q(s, a)$ for encountered state S_t and chosen ac-

tion A_t, G_t is the computed return (average of rewards) after time t, and α_t is a step-size parameter which is called the *learning rate* of the algorithm. This is what is usually done in Monte Carlo RL algorithms, and it requires that a complete list (or *episode*) $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, \ldots, S_{T-1}, A_{T-1}, R_T$ is available in order to compute G_t. Conversely, TD algorithms simplify this aspect by considering only the immediate reward and the value of subsequent experienced states and actions, in order to update the current estimate of the action-value function. In particular, the first TD algorithm considered in this work, SARSA, uses the following update rule

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha_t \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right], \tag{9}$$

where actions at each time step t are chosen using an *ε-greedy* policy with respect to Q, that is, $A_t \in \arg\max_{a \in \mathcal{A}(S_t)} Q(S_t, a)$ with probability $1 - \varepsilon_t$, and a is randomly chosen in $\mathcal{A}(S_t)$ with probability ε_t, being $0 < \varepsilon_t << 1$ a step-size parameter which expresses the balance between exploration and exploitation in the considered algorithms. The second TD algorithm we considered here is QL, which uses the slightly different update rule

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha_t \left[R_{t+1} + \gamma \max_{a \in \mathcal{A}(S_{t+1})} Q(S_{t+1}, a) - Q(S_t, A_t)\right]. \tag{10}$$

Since in QL actions are again selected according to an *ε-greedy* policy with respect to Q, the difference between it and SARSA is that the former is an *on-policy* control algorithm, i.e. the estimates of Q are updated using a correction factor given by actions selected using the current policy. On the contrary, QL is an *off-policy* control algorithm, meaning that the correction factor is determined using an action possibly different from the actual chosen one.

It is possible to prove that under suitable assumption both algorithms converge to an optimal action-value function, provided that all state-action pairs are visited an infinite number of times and suitable conditions on the parameters α_t and ε_t are satisfied (see [20] for technical details). However, the first requirement is hardly fulfilled when dealing with large state spaces. Indeed, the real bounds on computational resources at disposal, in terms of memory needed to store large arrays of $Q(s, a)$ values and of time needed to evaluate them, prevent from a direct application of the so called tabular version of these two algorithms. In order to tackle this issue, an alternative approach is the one provided by generalization from experience through the use of approximations $\hat{q}(s, a, \mathbf{w})$ of action-value functions, where $\mathbf{w} \in \mathbb{R}^d$ is a vector of weights which becomes the vector of unknowns of an optimization procedure.

Indeed, the *Mean Squared Value Error* to assess the approximation $\hat{q}(s,a,\mathbf{w})$ of the true action-value function $q_\pi(s,a)$ of a given policy π, can be defined as

$$\overline{VE}(\mathbf{w}) \equiv \sum_{s\in\mathcal{S}} \mu(s) \sum_{a\in\mathcal{A}(s)} \pi(a|s) \left[q_\pi(s,a) - \hat{q}(s,a,\mathbf{w})\right]^2, \tag{11}$$

where $\mu(s)$ is some distribution (so that $\mu(s) \geq 0, \sum_s \mu(s) = 1$) over $\mathcal{S}$. RL approximate solution methods aim to find a good local minimum $\mathbf{w}^*$ for $\overline{VE}(\mathbf{w})$ by using *Stochastic Gradient-Descent* methods (SGD), whose update rule at each time step is

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\right] \nabla_{\mathbf{w}} \hat{q}(S_t, A_t, \mathbf{w}_t), \tag{12}$$

where U_t is the current estimate of $q_\pi(S_t, A_t)$ and $\alpha \left[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\right]$ represents the step-size. Convergence properties for SGD depend on the kind of estimator U_t, on the type of the functional approximator $\hat{q}(s,a,\mathbf{w})$ used, and on the fact that the control algorithm used is an on-policy or off-policy one (see [3]). In this work we will consider the linear function approximation, that is $\hat{q}(s,a,\mathbf{w}) = \mathbf{w}^\intercal \mathbf{x}(s,a) = \sum_{i=1}^d w_i x_i(s,a)$, where $x_i : \mathcal{S} \to \mathbb{R}$ are called *feature representations* of state-action couples (s,a). In this case clearly $\nabla_{\mathbf{w}} \hat{q}(s,a,\mathbf{w}) = \mathbf{x}(s,a)$, so that combining (12) with the idea in SARSA updating rule (9), we obtain

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha \left[R_{t+1} + \gamma \mathbf{w}_t^\intercal \mathbf{x}(S_{t+1}, A_{t+1}) - \mathbf{w}_t^\intercal \mathbf{x}(S_t, A_t)\right] \mathbf{x}(S_t, A_t) \tag{13}$$

and analogously

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha \left[R_{t+1} + \gamma \max_{a\in\mathcal{A}(S_{t+1})} \mathbf{w}_t^\intercal \mathbf{x}(S_{t+1}, a) - \mathbf{w}_t^\intercal \mathbf{x}(S_t, A_t)\right] \mathbf{x}(S_t, A_t) \tag{14}$$

for QL.

It is possible to prove (see [24]) that, assuming technical conditions on the rewards, features, and the iterative decrease of the step-size parameters, the iteration (13) converges to a fixed point $\widetilde{\mathbf{w}}$ where $\overline{VE}(\widetilde{\mathbf{w}})$ is within a bounded expansion of the lowest possible error. Unfortunately, the same theoretical results do not hold in general for off-policy control algorithms with linear approximation, since divergence of the norm associated with the vector of weights could occur (see [2]). For these reasons recently a new algorithm has been introduced, namely Greedy-GQ, which is a generalization of QL using the linear approximation case and keeps the desired convergence properties. This algorithm needs to store values for an additional sequence of weights $\boldsymbol{\theta}_t \in \mathbb{R}^d$ and an additional step-size

parameter β_t, so that its updating rules are the following

$$\begin{aligned}
\delta_{t+1} &\leftarrow \left[R_{t+1} + \gamma \mathbf{w}_t^\intercal \mathbf{x}(S_{t+1}, a') - \mathbf{w}_t^\intercal \mathbf{x}(S_t, A_t)\right] && (15)\\
\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \alpha_t \left[\delta_{t+1} \mathbf{x}(S_t, A_t) - \gamma \boldsymbol{\theta}_t^\intercal \mathbf{x}(S_t, A_t)\right] \mathbf{x}(S_{t+1}, a') && (16)\\
\boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t + \beta \left[\delta_{t+1} - \boldsymbol{\theta}_t^\intercal \mathbf{x}(S_t, A_t)\right] \mathbf{x}(S_t, A_t) && (17)
\end{aligned}$$

with $a' \in \arg\max_{a \in \mathcal{A}(S_{t+1})} \mathbf{w}_t^\intercal \mathbf{x}(S_{t+1}, a)$. Note that by setting $\boldsymbol{\theta}_0 = \mathbf{0}$ and $\beta_t = 0$ the algorithm (15)-(17) reduces to QL.

3 Algorithms and Trading System

In this section we present the details of the applications of the three algorithms described in the previous section, to the development of automatic trading systems operating on Italian FTSE Mib stock market. The source of the data we used is the Bloomberg© database (see [7]), from which we collected daily close prices for five major companies (Enel, Generali, Intesa, Tim, Unicredit) from January 2000 to October 2018. Our aim is to improve the results obtained in [8] while keeping a similar simple structure of both the state space representing the stock market and the trading actions available.

Then we assume that at every time step t the trading system can invest all of its current capital opening or keeping a short or long position on a single stock, or it can close it and stay out of the market. This is formalized by setting $\mathcal{A}(\mathcal{S}_t) = \mathcal{A} = \{-1, 0, 1\}$ for each time t and each state S_t. Actions are chosen according to an *ε-greedy* policy with respect to the current approximation of the action-value function for the selected algorithm, with random ones picked from an uniform probability distribution on $\mathcal{A}$.

Feature representation of environmental states is a crucial topic for RL linear methods and can have a relevant impact on the efficiency of the algorithm. In this contribution we generalize the approach used in [8] by considering features not only for a given number n of past logarithmic returns of the considered stock price, but also for the current performance of the trade in action.

Formally, we first consider the vector $\mathbf{y}(S_t, A_t) \in \mathbb{R}^{n+1}$ define by

$$\begin{aligned}
y_i(S_t, A_t) &= \phi\left(\ln\left(\frac{P_{t-n+i}}{P_{t-(n+1)+i}}\right)\right), \quad \text{for } i = 1, \ldots, n && (18)\\
y_{n+1}(S_t, A_t) &= \phi(PL_t) && (19)
\end{aligned}$$

where $PL_t = 0$ if $A_{t-1} = 0$, otherwise it is the logarithmic return of the current trade, and

$\phi(x)$ is the same real-valued logistic function used in [8], that is

$$\phi(x) = \frac{a}{1 + be^{-cx}} - d. \tag{20}$$

Then, for the actual feature vector $\mathbf{x}(S_t, A_t)$ we adopt a block representation commonly used in RL algorithms (see [10]). That is, the vector $\mathbf{y}(S_t, A_t)$ is copied to one of the three slots of a zero vector with $|\mathcal{A}| \cdot (n+1) = 3 \cdot (n+1)$ elements according to the following rule:

$$\mathbf{x}(S_t, A_t) = \begin{cases} \left[\mathbf{y}(S_t, A_t) \quad \mathbf{0}^{n+1} \quad \mathbf{0}^{n+1}\right]^\intercal, & \text{if } A_t = -1 \\ \left[\mathbf{0}^{n+1} \quad \mathbf{y}(S_t, A_t) \quad \mathbf{0}^{n+1}\right]^\intercal, & \text{if } A_t = 0 \\ \left[\mathbf{0}^{n+1} \quad \mathbf{0}^{n+1} \quad \mathbf{y}(S_t, A_t)\right]^\intercal, & \text{if } A_t = 1 \end{cases} \tag{21}$$

where $\mathbf{0}^{n+1}$ is the null vector in $\mathbb{R}^{n+1}$.

For the reward R_{t+1} we considered two choices; the first one, as in [8] is

$$R_{t+1} = \frac{\mu(g_{l,t+1})}{\sigma(g_{l,t+1})} \qquad \text{Sharpe Ratio} \tag{22}$$

where μ and σ are respectively the sample mean and standard deviation of the rewards in the last l trading days.

The second one is

$$R_{t+1} = \frac{\mu(g_{l,t+1})}{1 + \max DD_{l,t+1}} \qquad \text{Calmar Ratio} \tag{23}$$

where $\max DD_{l,t+1}$ is the maximum drawdown, that is the difference between the maximum value of the equity gained by the trading system in the last l trading days and the subsequent minimum value.

Before going into the detailed presentation of the actual implementation of the used algorithms and of the results obtained, we want to highlight some differences of our modeling with respect to the one provided in [8]. First, one possible main drawback of the latter is that the average annual number of transactions of the algorithms is quite low. This could be related to the relative importance given to weight of the current trading action by the special form on linear approximation used in that contribution. Our choice (21) aims to a greater flexibility with respect to this perspective.

Second, in all our rests we used the same neutral initialization of the action-value function: that is we set the weights $\mathbf{w} = \mathbf{0}$ for the linear version of SARSA, QL and Greedy-GQ algorithms (13)-(17). This is another difference as compared to the approach used in [8], where a random initialization has been used. Indeed, we believe that on one hand the choice

of ε-greedy actions is enough to introduce a significant diversity between the behavior of algorithms in each run, which could be then exploited in order to select a single efficient trading system. On the other hand, the random initialization is a source of some kind of supervised information which could bias the algorithms towards some possibly not optimal policy.

4 Data and results

A preliminary analysis on the data provided by Bloomberg© database has been conducted by comparing them with the ones provided by another source of information (`http://www.investing.com`) in order to exclude missing, duplicated or incorrect prices from the time series, since this appears to negatively influence the performance of the used RL algorithms.

To keep into account transaction costs required for opening and closing each position, we considered them as a percentage rate of 0.15% that diminishes the return obtained by the current trading operation. As for the values of the parameters of the logistic transformation (20), we set them to $a = 2, b = 1, c = 10^2, d = 1$ after some preliminary tests regarding the impact of the value of c on the steepness of $\phi(x)$.

We then did a first analysis of the performance of the algorithms by running several simulations for each algorithm in order to compare their performance with respect to the choice of the step-size parameters, $\alpha, \beta, \gamma, \varepsilon$. More specifically, we analyzed the difference in the performance between setting them constant or decreasing over time according to the required conditions to ensure the convergence of the algorithms. Indeed it is reasonable to assume that the rewards in a noisy environment, as the stock market is, do not derive from a stationary probability distribution. In this case it could be argued that there does not exist a fixed optimal policy, instead it is changing over time, and the same apply obviously to optimal action-value functions. Consequently, the algorithm needs to go on performing exploratory actions (that means keeping ε_t constant) and learning and modifying action-value functions (that means keeping α_t and β_t constant).

So, we first considered several possible values for $\alpha = \beta \in \{0.05, 0.1, 0.15, 0.25, 0.1\}$, $\gamma \in \{0.5, 0.75, 0.975, 1\}$, $\varepsilon \in \{0.05, 0.1, 0.25, 0.5\}$, keeping fixed the values for $n = 5$ and $l = 5$ and we performed $N = 1000$ simulations for each combination of them and each algorithm with the two reward metrics (22)-(23). Then, we selected the values of the parameters that produce on average the best final equity value, namely $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05$. Furthermore we compared the results obtained using this setting with the ones obtained by imposing convergence-required decreasing values, that is $\alpha_t = \beta_t = \frac{1}{t}, \varepsilon_t = \frac{0.5}{t}$. The results are shown in Table 1 in terms of the ratio between averaged annual returns in the

continuous learning algorithms and the convergent ones. Is it remarkable that we always get best results with the constant choice of the step-size parameters, which confirms the non-stationary based hypothesis on the distribution of rewards. Moreover, we highlight that we have reported the result only for three of the considered stocks, since for the remaining two ones the average equity obtained with decreasing step-size parameters was lower then the initial capital. According to those results, we decided to adopt the continuous learning setting, with the above determined constant values of the step-size parameters in the subsequent analysis of the FTSs based on the three considered algorithms.

Table 1: Ratio between average annual returns in the continuous ($\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05$) learning and convergent ($\alpha_t = \beta_t = \frac{1}{t}, \varepsilon_t = \frac{0.5}{t}$) algorithms.

		Unicredit	**Intesa**	**Tim**
Sharpe	QL	3.33	1.55	1.74
	SARSA	3.06	1.43	1.66
	Greedy-GQ	4.32	2.32	2.22
Calmar	QL	3.15	1.65	2.05
	SARSA	2.85	1.65	1.98
	Greedy-GQ	4.51	2.47	2.75

One first question we have been able to get a positive answer for is the one related to the effective learning capability of the algorithms. In figures 1-10 we show some examples of the plots of respectively all and average equity lines for $N = 5000$ simulations of the algorithms applied to the different stocks with an initial capital $C = 100$. It is possible to see that, even with the presence of some periods with a drawdown and remarkable differences in the final value of equity across the considered stocks, after around $t = 2000$ time periods all the algorithms obtain a constantly increasing mean equity value. This behaviour can be explained as if the algorithms need a training period of around $t = 2000$ time periods, after which they effectively *learn* on average a sub-optimal policy.

Since however in every single simulation we get a possibly different policy, as the figures of the equity lines for each simulation show, we then considered the issue of selecting a single operational FTS on the basis of the different N actions selected at each time step t. The approach we adopted consists in computing before the average value $\overline{A_t} = \frac{\sum_{i=1}^{N} A_{i,t}}{N}$ of the actions taken at each time step in each run of the selected algorithm, and then to determine the operative action $\widetilde{A}_t$ on the basis that this value satisfies some criterion. In particular, we considered two possibilities. For the first one we used a threshold based criterion:

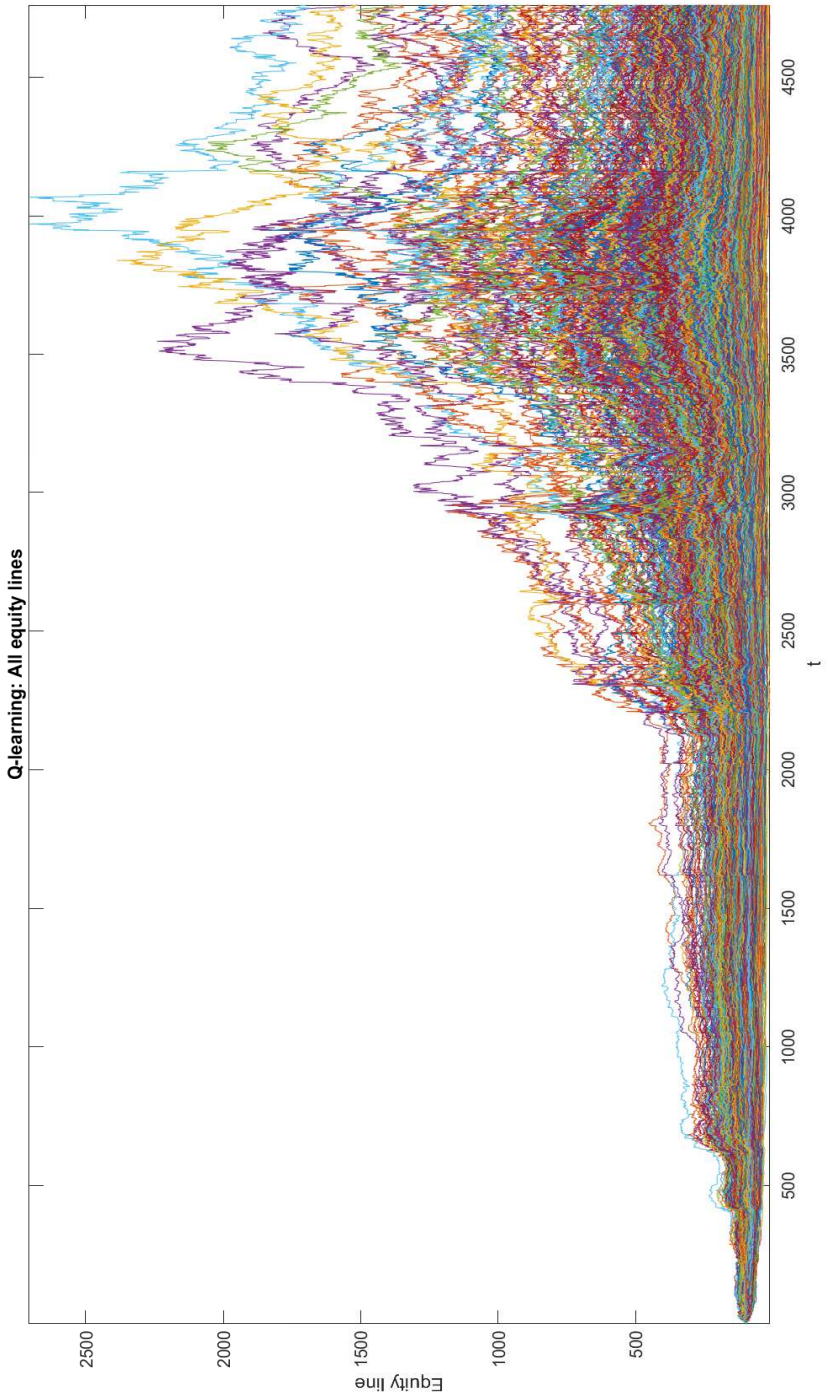


Figure 1: QL-based FTSs, Enel equity lines, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5, N = 5000$

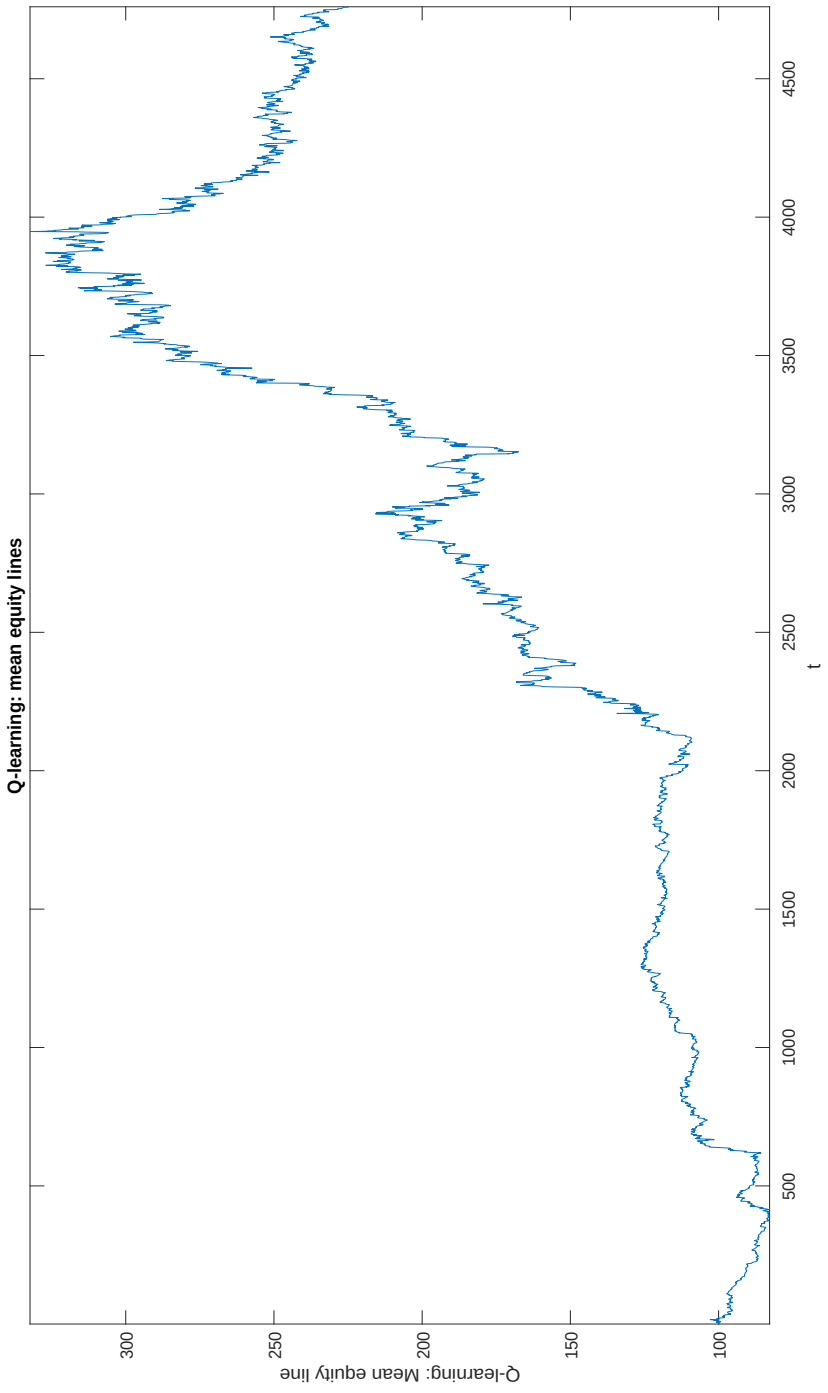


Figure 2: QL-based FTSs, Enel mean equity line, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5, N = 5000$

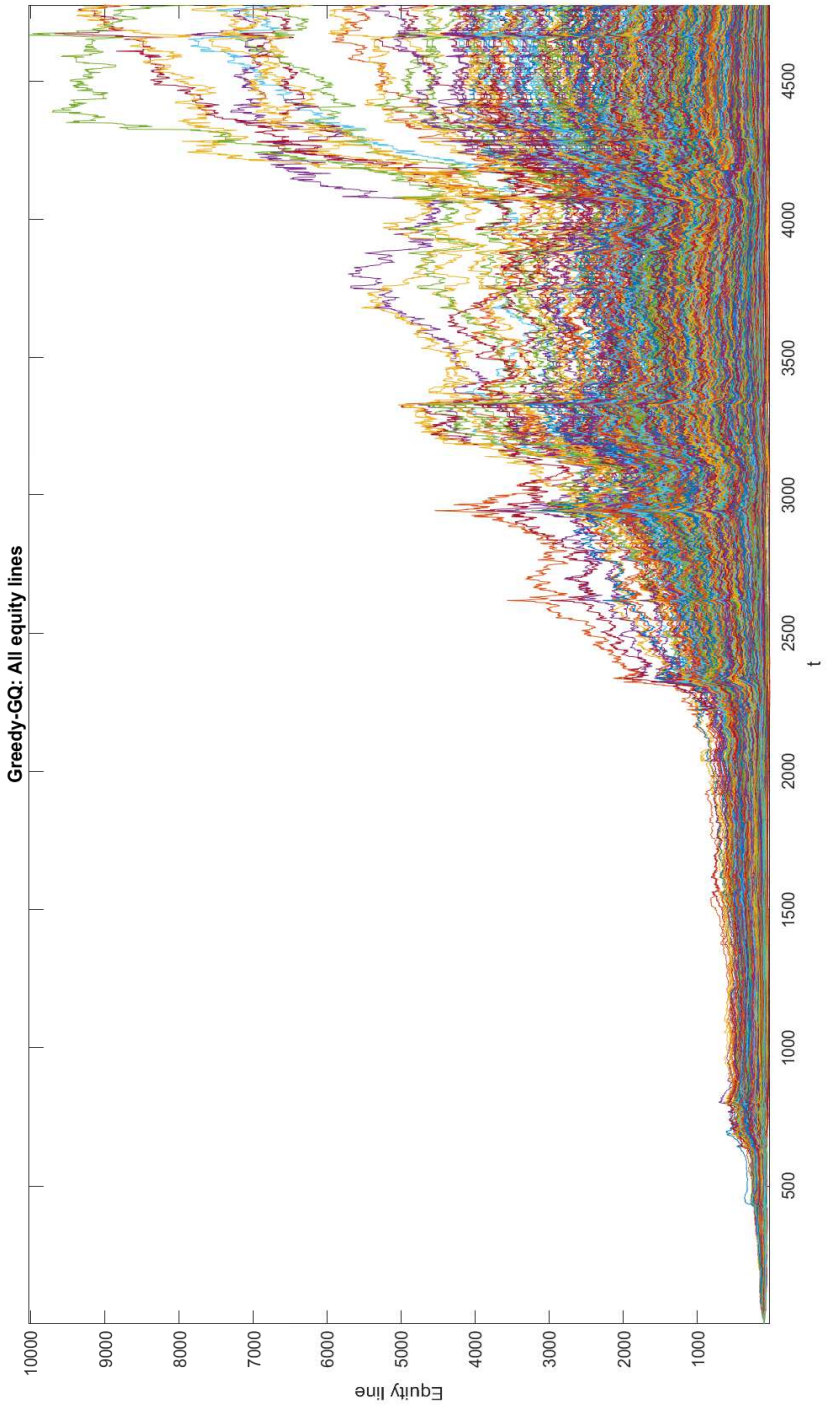


Figure 3: Greedy-GQ-based FTSs, Generali equity lines, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 5, l = 5, N = 5000$

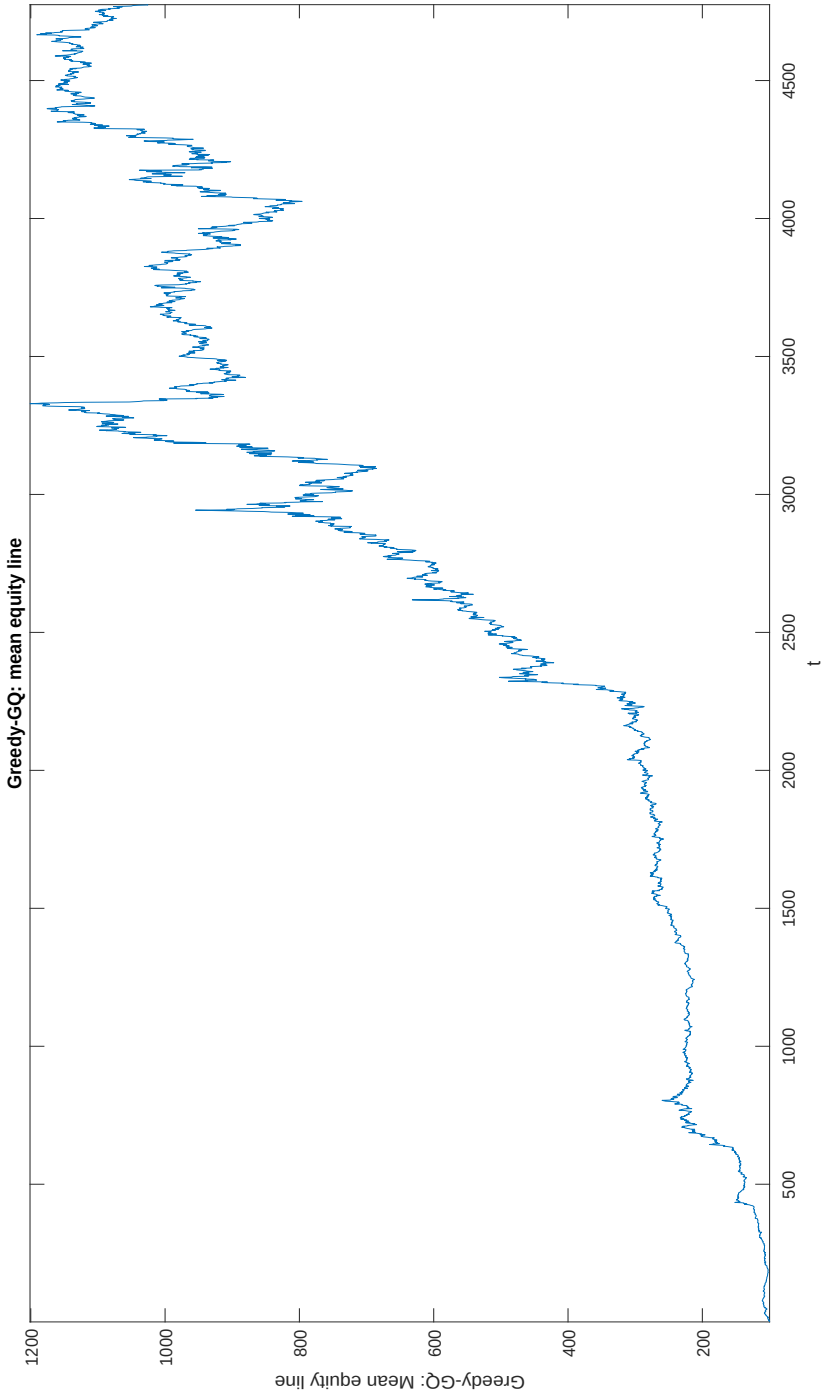


Figure 4: Greedy-GQ-based FTSs, Generali mean equity line, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 5, l = 5, N = 5000$

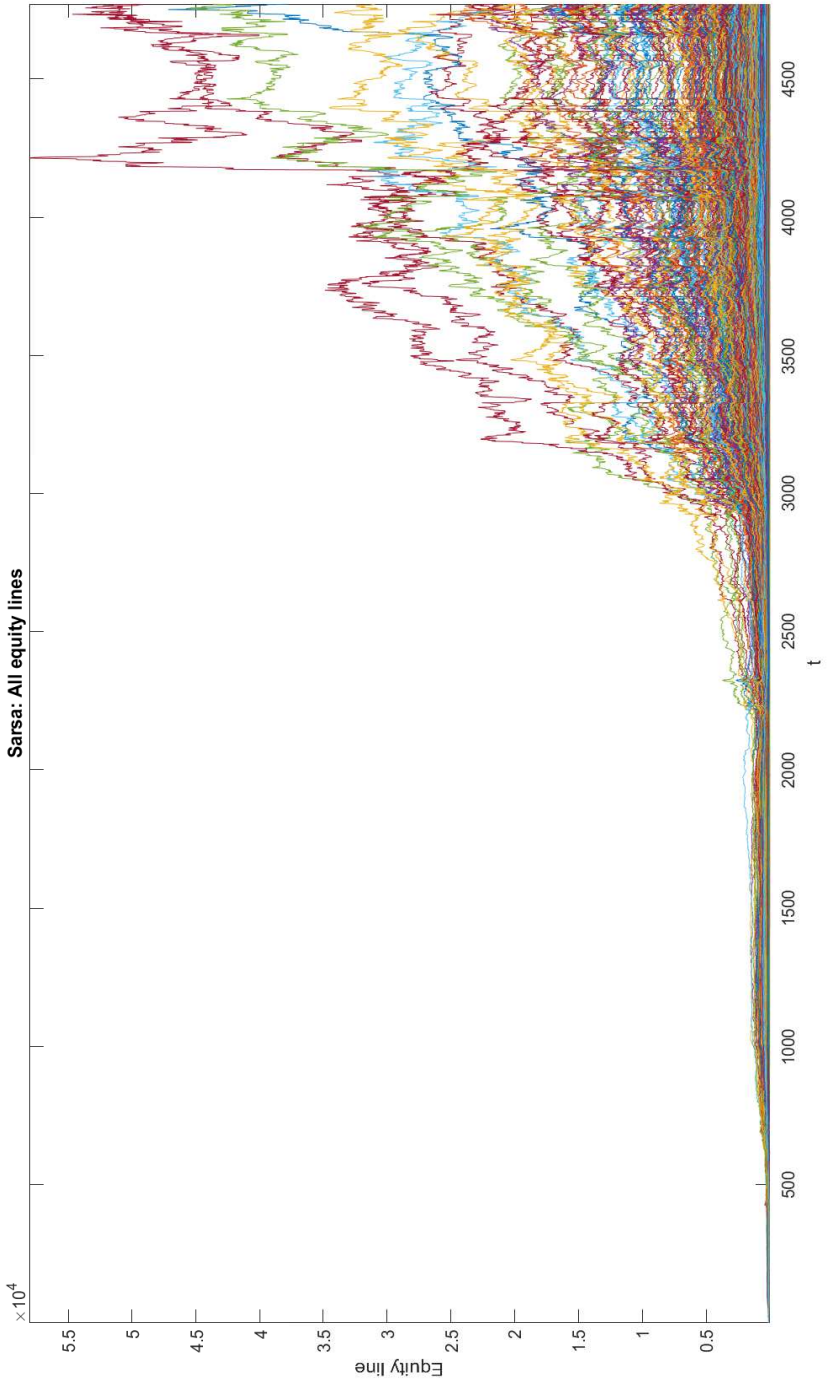


Figure 5: SARSA-based FTSs, Intesa equity lines, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 10, l = 5, N = 5000$

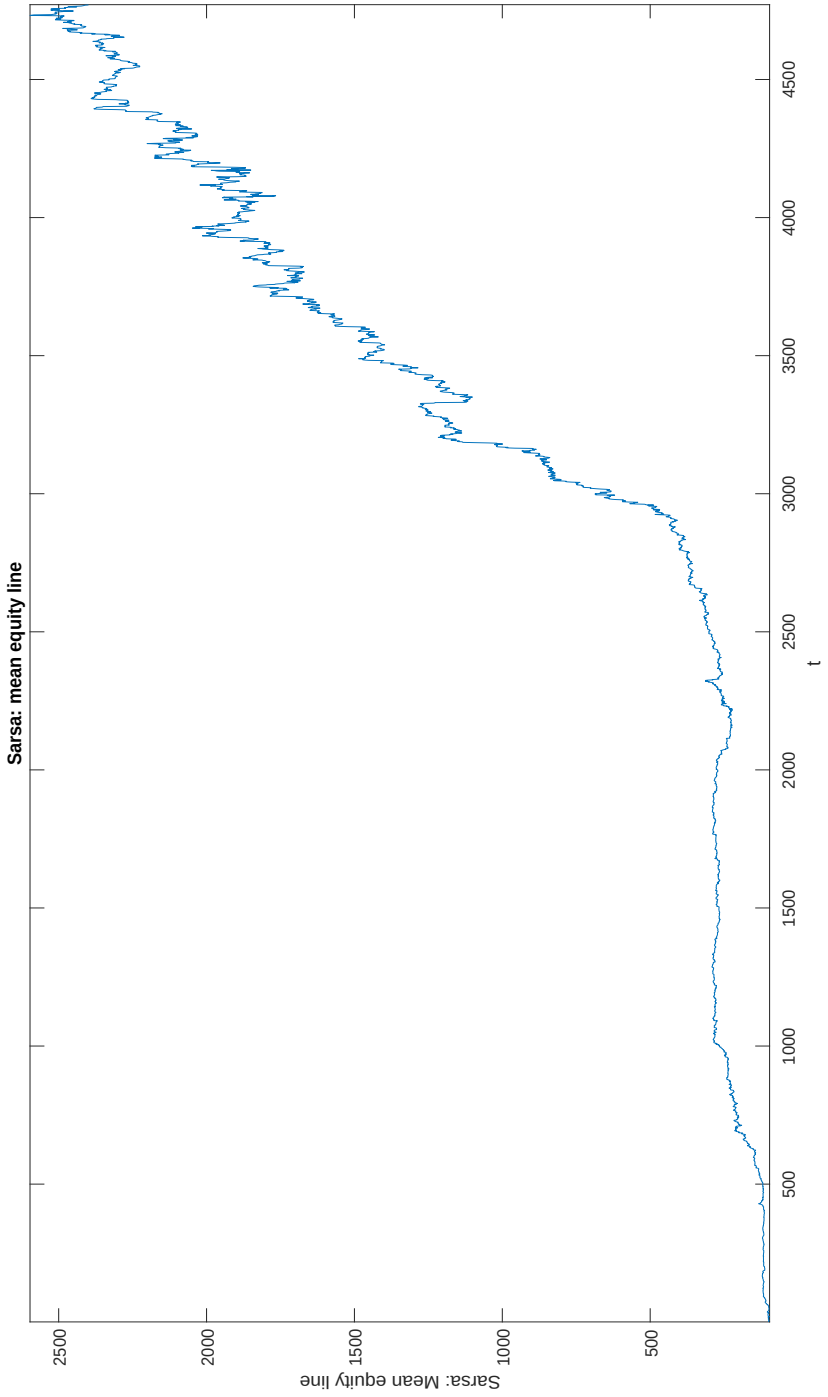


Figure 6: SARSA-based FTSs, Intesa mean equity line, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 10, l = 5, N = 5000$

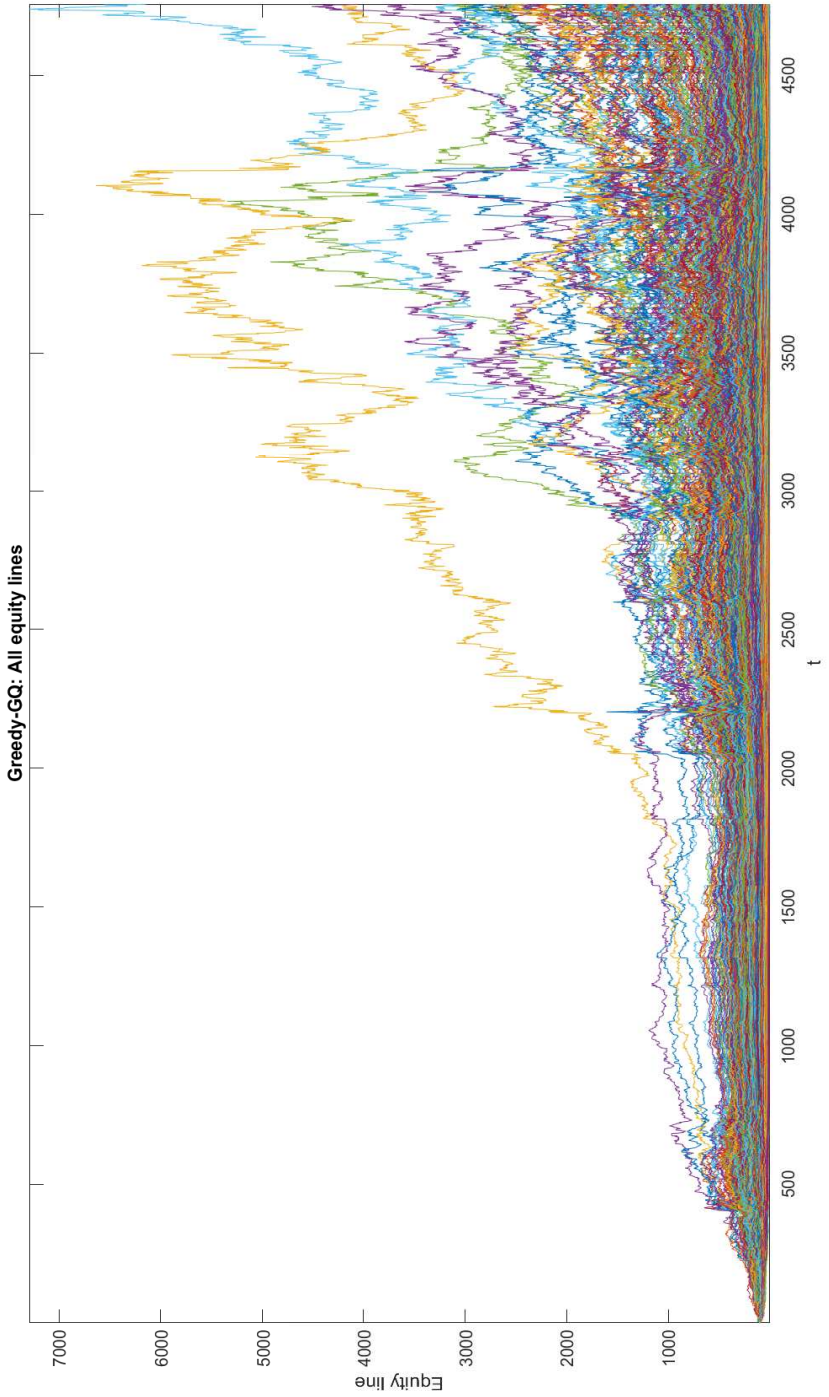


Figure 7: Greedy-GQ-based FTSs, Tim equity lines, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5, N = 5000$

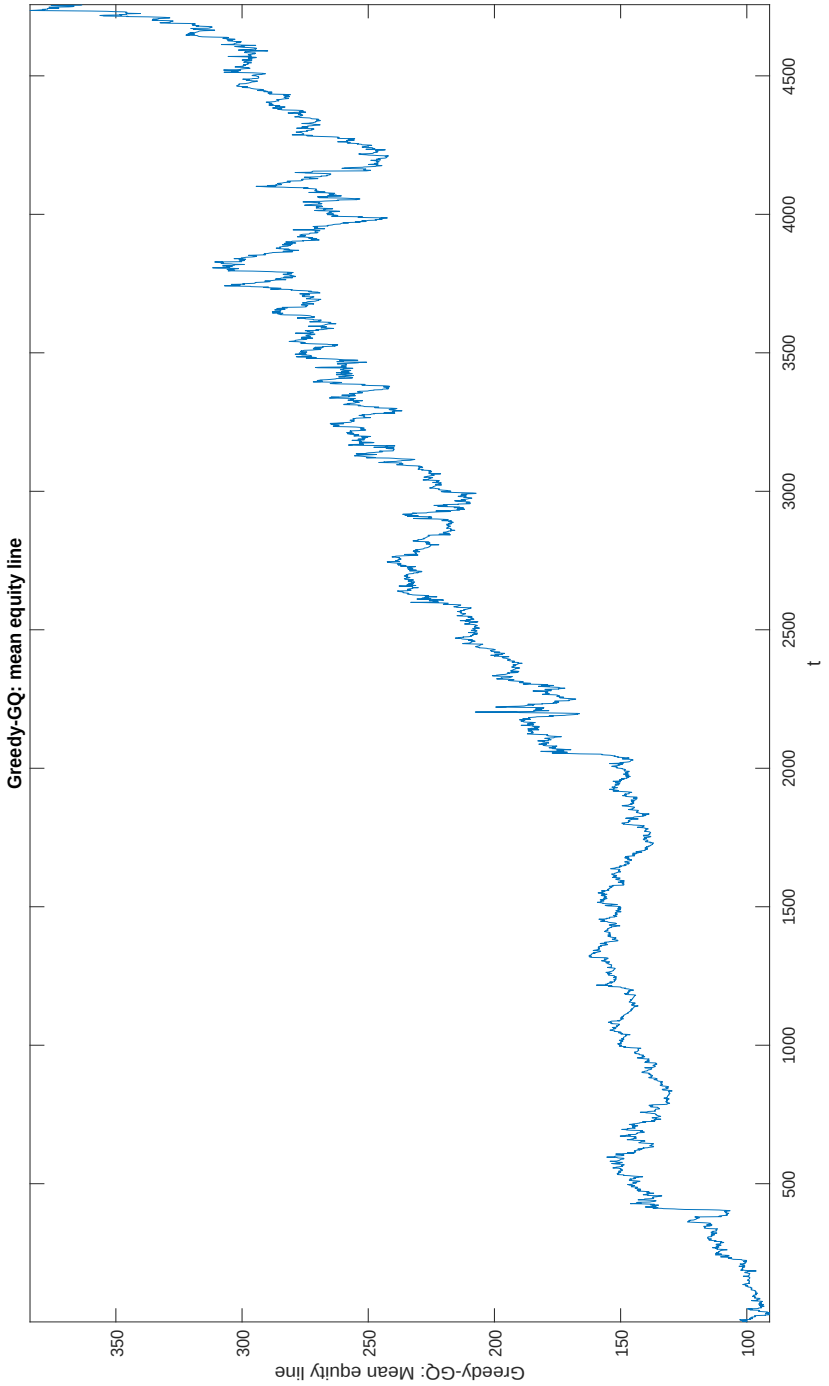


Figure 8: Greedy-GQ-based FTSs, Tim mean equity line, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5, N = 5000$

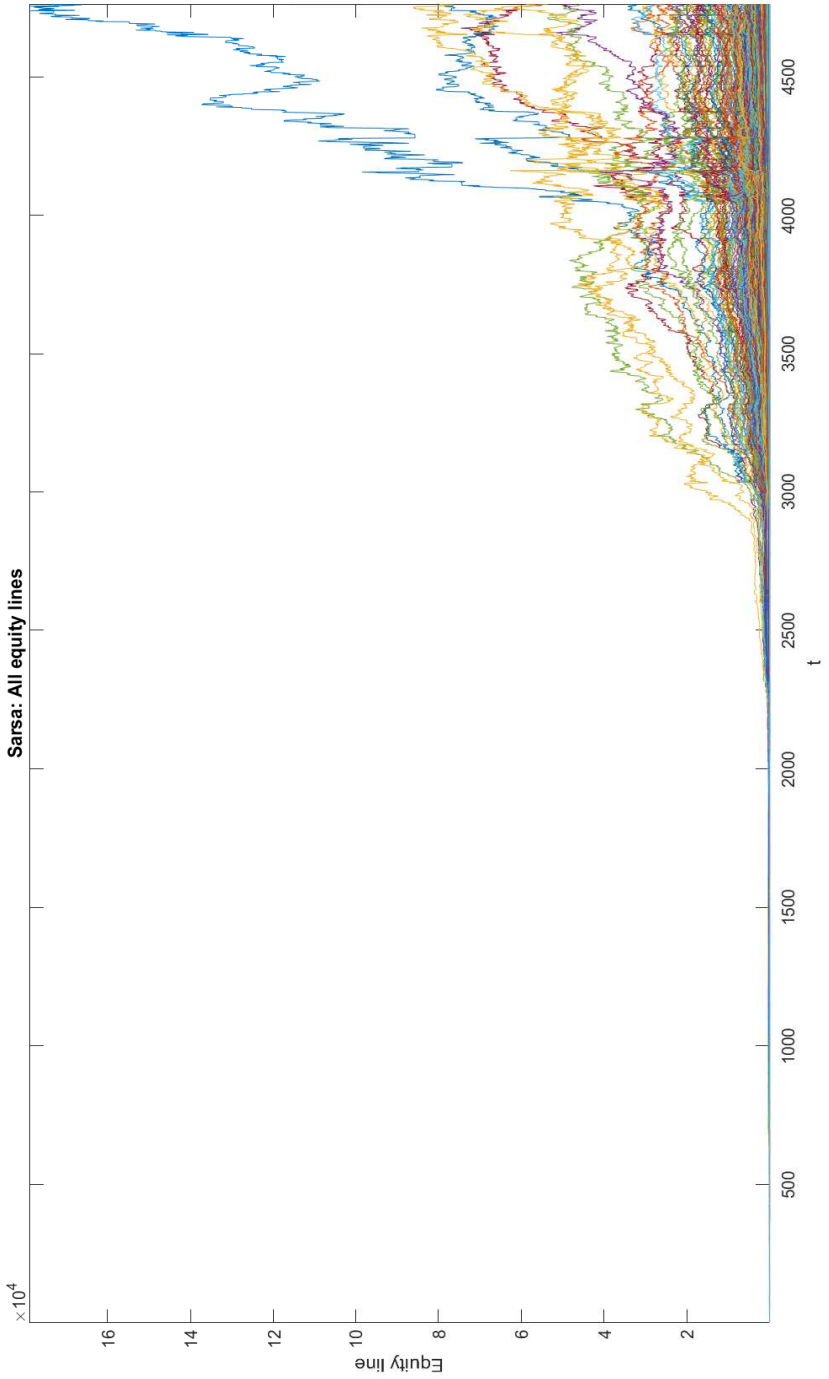


Figure 9: SARSA-based FTSs, Unicredit equity lines, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5, N = 5000$

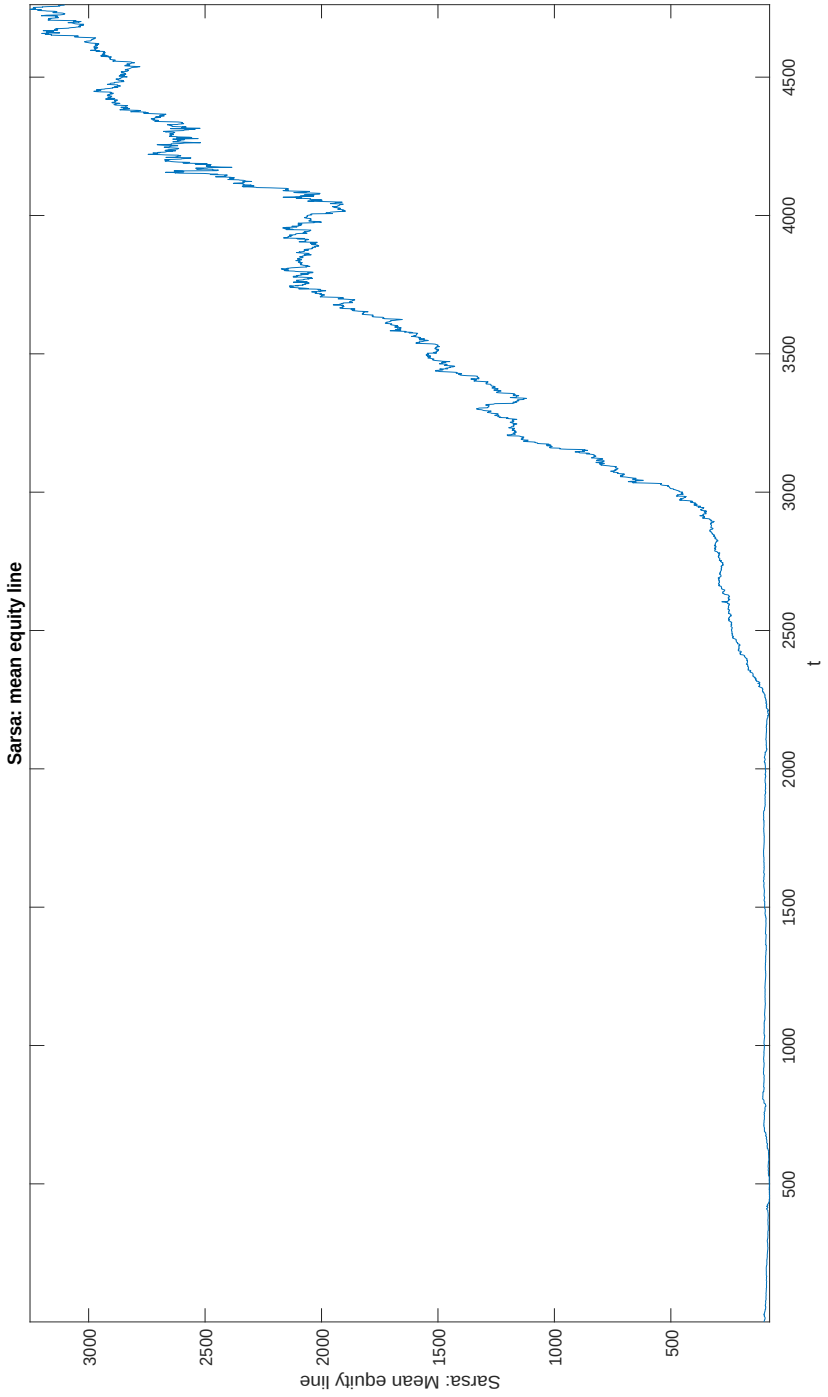


Figure 10: SARSA-based FTSs, Unicredit mean equity line, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5, N = 5000$

$$\widetilde{A}_t = \begin{cases} -1, & \text{if } \overline{A_t} \in [-1, -\tau) \\ 0, & \text{if } \overline{A_t} \in [-\tau, \tau] \\ 1, & \text{if } \overline{A_t} \in (\tau, 1] \end{cases} . \tag{24}$$

where $\tau \in (0, 1)$. Another criterion we adopted is the following one: first we employ a Student t-test of the null hypothesis that the sample data $\{A_{i,t}\}_{i=1}^{N}$ are from a population with mean equal to zero at a given significance level `alpha`. If the null hypothesis is accepted, then we set $\widetilde{A}_t = 0$, otherwise $\widetilde{A}_t = \text{sgn}(\overline{A_t})$. After some preliminary test to determine the best values for τ and `alpha`, we set them respectively to $\tau = 0.05$ and `alpha` $= 5\%$.

In Tables 2-11 we show the results of the applications of these two criteria to all considered stocks, using both the Sharpe and Calmar ratio reward metrics, for different values of the number of past returns n considered in the feature representation of the stock market state and of the past l days considered in the computation of the rewards. In every table for each algorithm we report the two values of the annual average return obtained by the correspondent FTS: that is in the first row there is the one obtained using the threshold based operative FTS, and in the second one the t-test based one. Although there are significant differences in the returns obtained for the different stocks, some general properties of the FTSs can be remarked. First of all, it can be observed that there isn't a reward metric that is certainly better for each combination of n, l, stock, and FTS, while it appears that the choice mostly depends on the considered stock.

Instead, with regards to the algorithms the FTS are based on, we notice that for each stock the maximum return is obtained using the Greedy-GQ algorithm, regardless of the reward metrics considered, except for the Enel S.p.A one where the best algorithm is the SARSA one. Then, even though, as we have seen from Table 1, convergence conditions on the step-size parameters don't give better performances, nevertheless FTSs based on the two algorithms for which a theoretical convergence property is established seems to behave better than QL based one. This is confirmed when comparing the performances of the FTSs for each stock and for each combination of reward metric, n, l, and operative criterion used: for each stock we have 24 possible outcomes, for a total of 120 ones. Indeed, SARSA based FTSs are better than QL ones in 89 occurrences, that corresponds to $74,16\%$, and Greedy-GQ based ones are better in 113 ones, that correspond to $94,17\%$.

Table 2: Enel S.p.A., Operative annual returns (%), Sharpe Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	10.64	7.64	12.94
		10.33	7.91	12.86
	10	9.33	9.48	16.59
		8.89	9.44	16.61
10	**5**	14.40	11.66	14.09
		15.18	11.56	14.26
	10	11.39	7.36	10.75
		11.97	7.17	10.70
20	**5**	18.04	14.56	15.16
		18.57	14.84	15.44
	10	12.99	7.78	13.78
		13.57	8.65	13.66

Table 3: Enel S.p.A., Operative annual returns (%), Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	13.18	9.94	12.20
		13.25	10.53	12.20
	10	16.14	15.78	16.12
		16.34	16.02	16.13
10	**5**	13.90	14.50	14.68
		14.21	13.92	15.28
	10	12.97	10.32	13.18
		12.46	11.07	13.22
20	**5**	20.35	17.72	18.08
		20.28	18.15	18.10
	10	12.46	9.38	15.53
		12.46	9.11	15.68

As for the choice of best values for n and l, we notice that the maximum return is obtained in 7 out of 10 possible times using $n = 20$, so it appears that, despite the increase in the dimension of the state space and the consequent impact on the approximation effort, the more information about past returns is used, the best performance is achieved. This possibly suggests that several past trading days are needed in order to obtain a representation of the state space for which Assumption 2.1 is satisfied. There is instead not a clear rule for the choice of the best value of l, even if it appears that when using a shorter history for the

returns of the stocks ($n = 5, 10$), better results are obtained using $l = 10$: this can possibly be seen as a sort of compensation between the information provided to the agent from states and rewards. Finally, we remark that there are not statistically significant differences between the performances obtained by FTSs based on the threshold criterion and the ones based on the t-test.

Table 4: Generali S.p.A., Operative annual returns (%), Sharpe Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	20.15	14.74	20.80
		20.37	15.20	20.72
	10	19.67	14.93	21.31
		19.50	14.60	21.58
10	**5**	20.03	19.26	21.18
		19.72	18.87	20.65
	10	18.91	9.72	14.82
		19.02	9.74	14.57
20	**5**	16.53	22.73	23.31
		16.06	22.41	23.59
	10	10.84	13.11	23.44
		10.36	13.67	23.91

Table 5: Generali S.p.A., Operative annual returns (%), Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	19.69	21.02	26.67
		19.32	21.21	26.16
	10	18.78	18.30	26.63
		18.78	19.57	26.64
10	**5**	24.07	21.55	20.70
		24.22	21.36	20.59
	10	19.01	10.28	18.91
		18.43	10.45	18.75
20	**5**	11.94	18.68	22.26
		13.39	18.24	21.73
	10	14.30	16.41	21.23
		13.90	16.94	21.98

Table 6: Intesa S.p.A., Operative annual returns (%), Sharpe Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	l	SARSA	QL	Greedy-GQ
5	**5**	48,98	42,59	46,87
		49,83	42,77	46,81
	10	40,76	43,60	49,31
		40,84	43,44	49,17
10	**5**	50,45	48,71	54,94
		50,13	48,00	54,50
	10	37,95	31,45	47,10
		37,89	31,13	46,90
20	**5**	33,76	42,25	49,65
		34,25	41,27	49,84
	10	26,82	33,70	39,06
		25,87	33,74	38,48

Table 7: Intesa S.p.A., Operative annual returns (%), Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	l	SARSA	QL	Greedy-GQ
5	**5**	49.37	44.26	49.50
		49.01	45.02	50.07
	10	50.72	42.49	51.49
		51.14	41.98	50.74
10	**5**	45.68	43.10	48.64
		46.45	43.56	48.73
	10	38.34	30.55	41.79
		39.38	29.89	41.93
20	**5**	32.13	37.90	42.69
		32.55	37.00	42.32
	10	25.00	34.92	35.61
		24.86	35.84	35.09

In Figures 11-13 we show some examples of the behaviour of the operative FTSs. Each figure is composed of three panels: in the first one there is the daily end-of-day price of the considered stock, in the second one there is the action undertaken by the FTS, and in the third one there are both the gross (that is, without considering transaction costs), and the net equity line obtained by the FTS. It can be seen that the FTSs are in general performing a large number of operations,[1] and this reflects on the fact that there is a large difference

[1] A switch of position from long to short and vice versa is counted as two operations.

in plots between gross and net equity obtained by the FTSs. This is a clear difference with respect to the behaviour of the FTSs analyzed in [8], where the average annual number of operations was quite low, and confirms the greater flexibility of the choice (21) of the linear approximation adopted in this contribution.

Table 8: Tim S.p.A., Operative annual returns (%), Sharpe Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	21.56	20.14	21.01
		22.09	20.17	20.96
	10	20.76	15.34	27.59
		20.75	15.71	27.40
10	**5**	20.16	13.80	22.31
		19.63	13.61	22.40
	10	17.45	14.11	22.73
		16.37	13.65	22.02
20	**5**	17.73	8.73	32.22
		18.60	8.95	32.58
	10	14.49	7.76	30.58
		13.88	7.38	30.60

Table 9: Tim S.p.A., Operative annual returns (%), Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	25.93	19.35	19.70
		26.28	19.71	19.71
	10	20.70	20.60	29.59
		20.63	20.85	29.52
10	**5**	19.54	14.80	23.98
		18.94	14.29	23.74
	10	17.56	20.03	21.89
		17.40	19.65	21.76
20	**5**	15.87	6.66	30.18
		16.42	7.04	31.27
	10	9.87	4.31	25.16
		10.66	4.28	25.58

As a final result, in table 12 we show the values of the maximal drawdown and of the effective equity Calmar ratio for the FTSs which achieved the best annual average return for each stock and for the two reward metrics (22) and (23). While one should expect that FTS

which use the Calmar ratio based reward metric should give lower drawdown and better effective Calmar ratio, it can be seen that in some cases this is not true, and generally the maximum drawdown for all the FTSs is quite high. This suggests that in RL framework the classical financial measures of risk should be considered with care when used as rewards metrics.

Table 10: Unicredit S.p.A., Operative annual returns (%), Sharpe Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	64.74	63.98	62.32
		64.76	63.08	62.99
	10	34.91	37.49	65.79
		36.99	36.05	65.81
10	**5**	60.11	58.43	68.77
		59.51	61.05	68.59
	10	46.20	35.56	61.43
		45.58	36.23	61.86
20	**5**	68.86	64.42	78.51
		69.29	64.61	79.51
	10	51.08	29.04	74.19
		50.33	29.96	75.06

Table 11: Unicredit S.p.A., Operative annual returns (%), Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, \tau = 0.05, \delta = 0.15\%, 5000$ tests

n	**l**	**SARSA**	**QL**	**Greedy-GQ**
5	**5**	60.11	64.61	64.58
		60.04	64.18	64.22
	10	52.69	48.69	62.86
		53.28	49.78	63.19
10	**5**	55.26	47.90	57.79
		55.90	48.07	58.70
	10	47.94	32.03	54.85
		47.49	33.26	54.78
20	**5**	75.33	69.34	74.04
		74.86	68.76	73.02
	10	62.04	42.68	75.58
		61.75	45.12	76.45

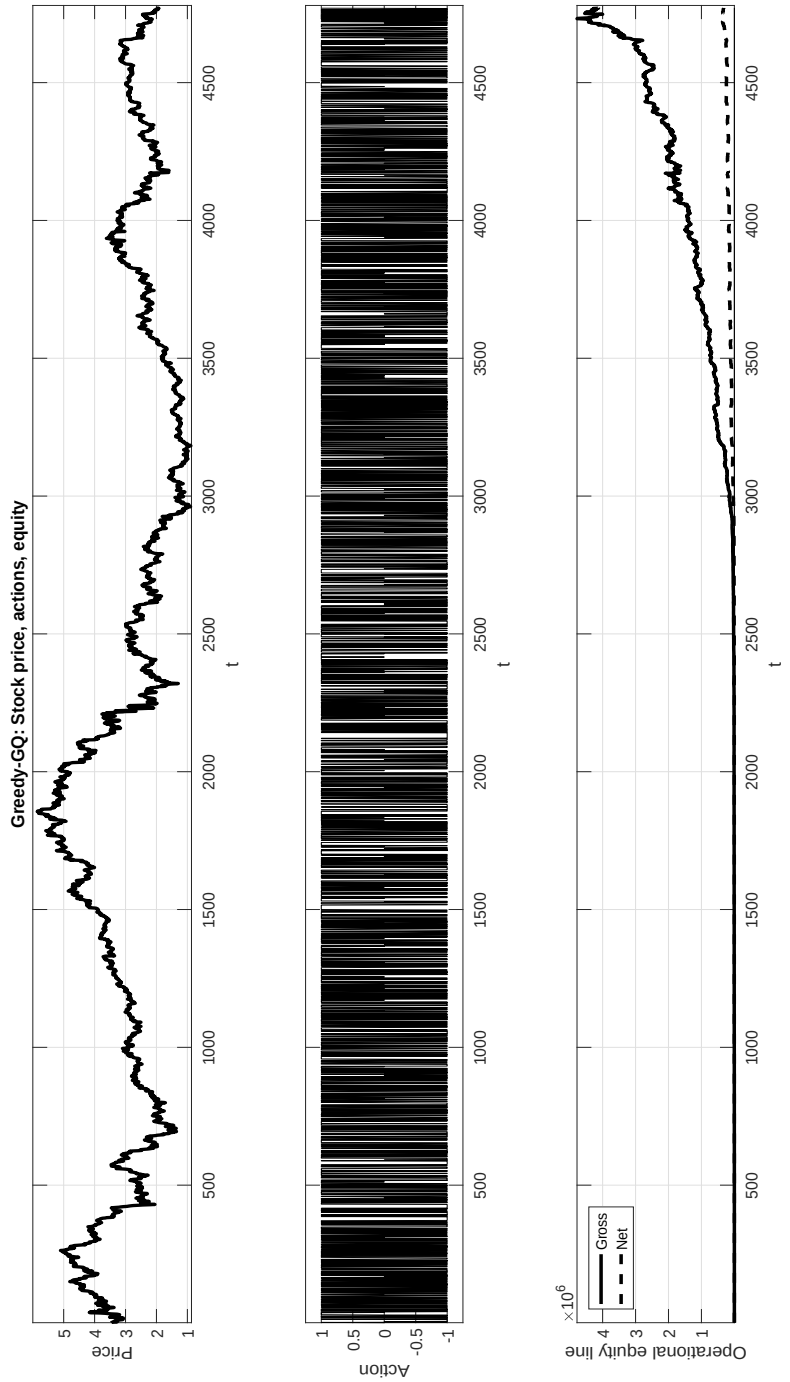


Figure 11: Greedy-GQ threshold-based FTS, Intesa S.p.A operative results, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 10, l = 5$

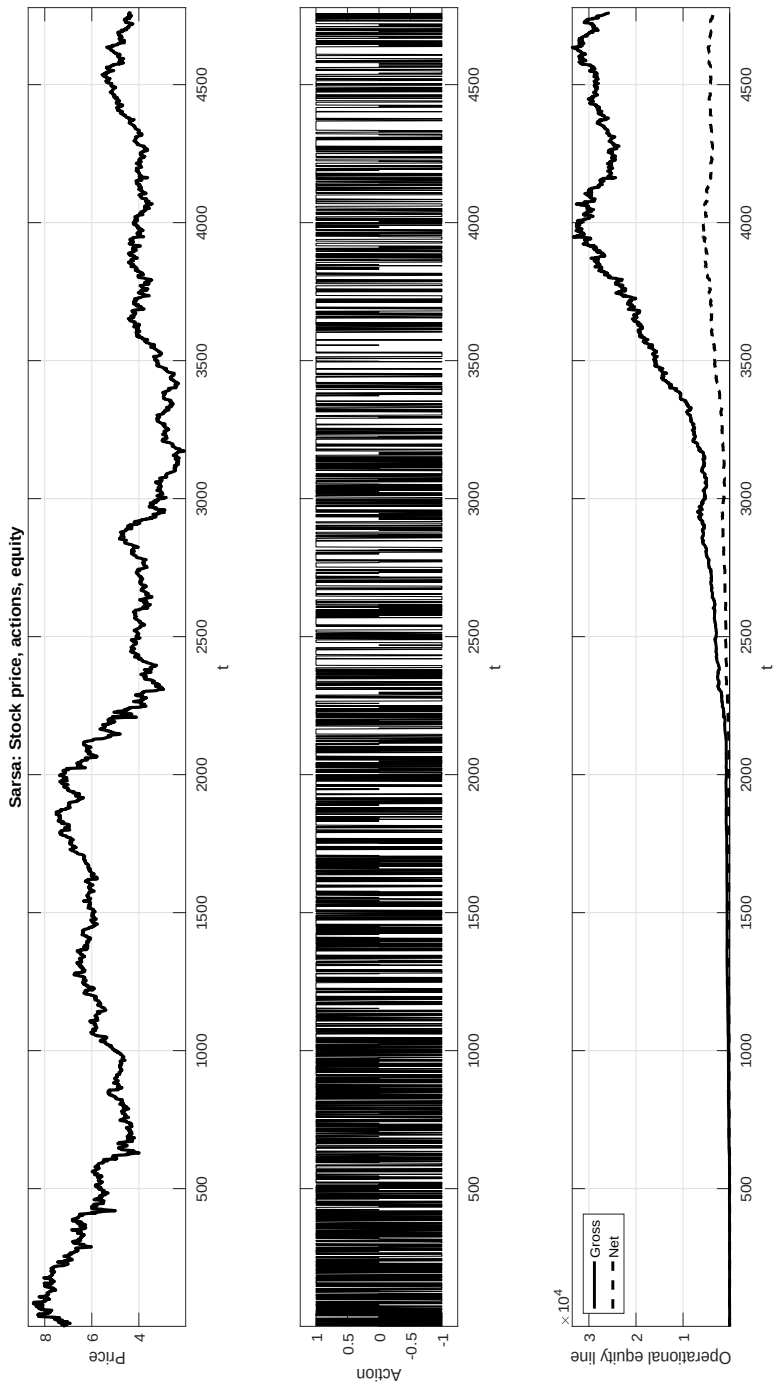


Figure 12: SARSA threshold-based FTS, Enel S.p.A operative results, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5$

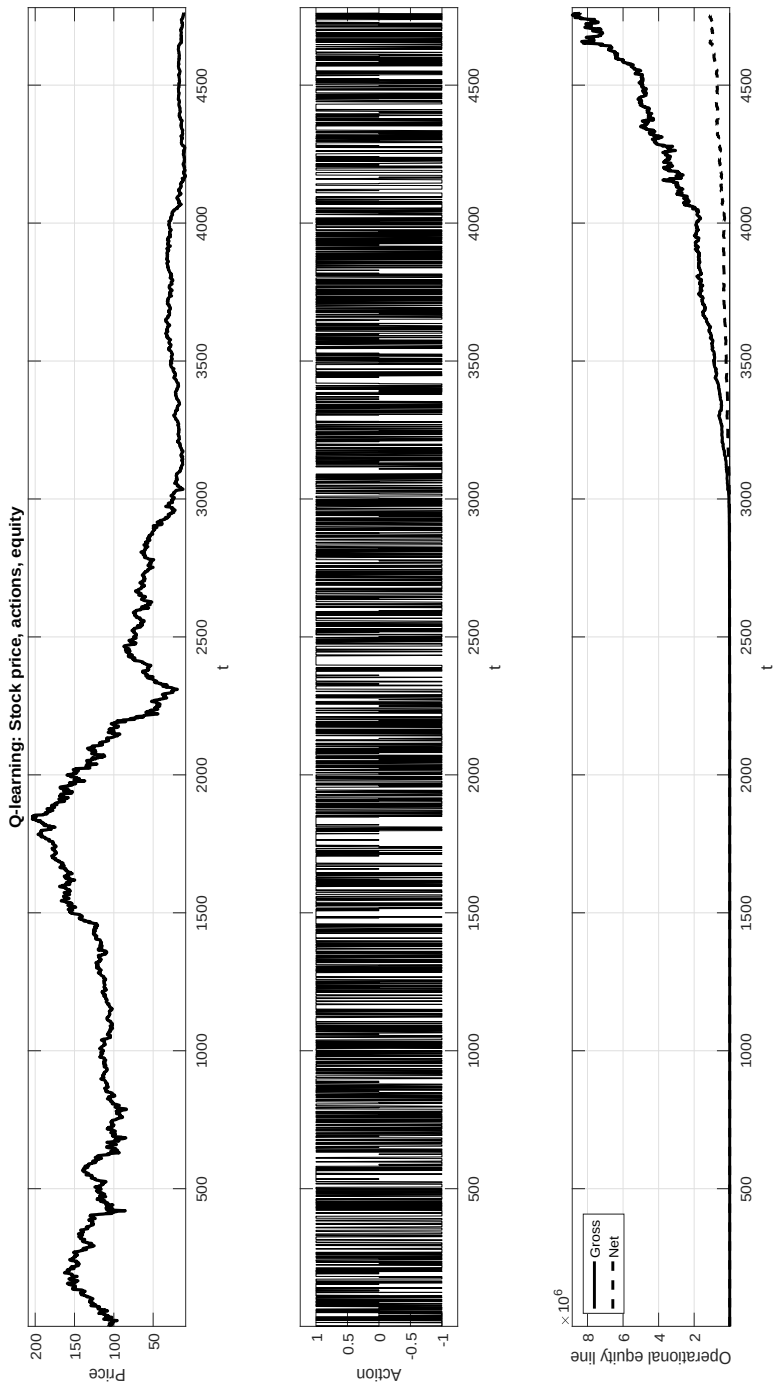


Figure 13: QL t-test-based FTS, Unicredit S.p.A operative results, Calmar Ratio, $\alpha = \beta = 0.05, \gamma = 0.975, \varepsilon = 0.05, n = 20, l = 5$

Table 12: Maximal drawdown (%) and Calmar Ratio for the best FTSs obtained with the two reward metrics for each stock

	Sharpe			**Calmar**		
Stock	**Return**	**maxDD**	**Calmar Ratio**	**Return**	**maxDD**	**Calmar Ratio**
Enel	18.57%	41,83%	0.44	20.35%	40.01%	0.51
Generali	23.91%	36.84%	0.65	26.67%	39.76%	0.67
Intesa	54.94%	38.22%	1.44	51.49%	43.89%	1.17
Tim	32.58%	30.82%	1.06	31.27%	36.79%	0.85
Unicredit	79.51%	42.07%	1.89	76.45%	35.38%	2.16

5 Conclusions and future works

All the FTSs presented in this contribution obtain positive results, especially in terms of the pure profitability, with differences mainly related to the specific stock they have been applied to. In particular, FTSs based on the Greedy-GQ algorithm show the best results in almost all considered cases. Generally, the results obtained show improvements with respect to the ones obtained in [8], but they also introduce some questions that require further investigation. The high number of annual operations carried out by the FTSs is a possible drawback in the case of an increase in the percentage transaction cost in order to compute also the effect of taxation in the profitability of the FTSs. Moreover, despite the high annual rewards obtained, the presence of large drawdowns is a concern for investors. In the future we aim to study the sensitivity of the performance of the FTSs with respect to change in the reward metrics, in order to minimize the two aforementioned drawbacks. Moreover, the role of the parameter c of the function $\phi(x)$, is another issue of interest for future investigation. Indeed, choosing high values for c from one hand allows to obtain a very steep increasing function, which increases the sensitivity of the algorithm to small differences in the returns of stock prices. On the other hand, this suggests that an alternative way to describe the state of the stock market could be the one that classifies every return as a member of a finite set. Then, if the cardinality of this set is not too high, a tabular version of the (9) or (10) could be more appropriate to solve the RL problem, and in the future we are interested in comparing results obtained with this approach to the present ones.

References

[1] Bahrammirzaee, A. (2010). A comparative survey of artificial intelligence applications in finance: Artificial neural networks, expert systems and hybrid intelligent systems, Neural Computing and Applications, 19(8), 1165–1195.

[2] Baird, L.C. (1995). Residual algorithms: Reinforcement learning with function approximation. Proceedings of the Twelfth International Conference on Machine Learning, 30–37. Morgan Kaufmann. [`http://www.leemon.com/papers/residual/residual.pdf`].

[3] Barto, A.G., Sutton, R.S. (2018). Reinforcement Learning: An Introduction. The MIT Press.

[4] Bellman, R.E. (1957). Dynamic programming. Princeton University Press.

[5] Bertsekas, D.P., Tsitsiklis, J.N. (1996). Neuro-Dynamic Programming. Athena Scientific.

[6] Bertsekas, D.P. (2012). Dynamic Programming and Optimal Control: Approximate Dynamic Programming. Athena Scientific. [Volume 2 – Fourth edition]

[7] Bloomberg Finance L.P. (2019). `https://www.bloomberg.com/professional/product/market-data/`.

[8] Corazza, M., Sangalli, A., (2015). Q-Learning and SARSA: a comparison between two intelligent stochastic control approaches for financial trading. Working Papers, Department of Economics, Ca' Foscari University of Venice, 15.

[9] Cuthbertson, K., Nitzsche, D. (2004). Quantitative Financial Economics. Wiley.

[10] Geramifard, D.C., How, J.P. (2013). Off-policy learning combined with automatic feature expansion for solving large MDPs. Proceedings of the 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making, 29–33. Princeton University Press.

[11] Gosavi, A. (2015). Simulation-Based Optimization. Parametric Optimization Techniques and Reinforcement Learning. Springer.

[12] Howard, R. (1960). Dynamic Programming and Markov Processes. MIT Press.

[13] Lo, A.W. (2004). The Adaptive Markets Hypothesis. Market efficiency from an evolutionary perspective. The Journal of Portfolio Management, 30(5), 15–29.

[14] Lo, A.W. (2005). Reconciling efficient markets with behavioral finance: The Adaptive Markets Hypothesis, The Journal of Investment Consulting. 7(2), 21–44.

[15] Lo, A. W. (2012). Adaptive markets and the new world order. Financial Analysts Journal, 68(2), 18–29.

[16] Lo, A. W. (2017). Adaptive Markets. Financial Evolution at the Speed of Thought. Princeton University Press.

[17] Maei, H.R., Szepesvári, C., Bhatnagar, S., Sutton R.S. (2010). Toward off-policy learning control with function approximation. International Conference on Machine Learning (ICML), 719–726. Omnipress.

[18] Ross, S. (1983). Introduction to Stochastic Dynamic Programming. Academic Press.

[19] Rummery, G.A., Niranjan, M. (1994). On-line Q-Learning using connectionist systems. Technical Report CUED/F-INFENG/TR, Engineering Department, Cambridge University, 166.

[20] Singh, S., Jaakkola, T., Littman, M.L., Szepesvri, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. Machine Learning, 38(3), 287–308.

[21] Singh, S.P., Sutton, R.S. (1996). Reinforcement learning with replacing eligibility traces. Machine Learning, 22(1-3), 123–158.

[22] Sutton, R.S. (1988). Learning to predict by the method of temporal differences. Machine Learning, 3(1), 9–44.

[23] Tsitsiklis, J.N., Van Roy, B. (1996). Feature-based methods for large scale dynamic programming. Machine Learning, 22(1-3), 59–94.

[24] Tsitsiklis, J.N., Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control, 42(5), 674-–690.

[25] Watkins, C.J.C.H., Dayan, P. (1992). Q-Learning. Machine Learning, 8(3–4), 279–292.