# ForSts: Tacit Collusion in the Repeated Non-Cooperative Games Using Forwarding N-Steps Reinforcement Learning Algorithm

**Amin Golzari Hormozi[1], Seyed Hossein Khasteh[1], Amirhossein Nikoofard[2], Zahra Shirmohammadi*[3]**

[1] Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran
golzari@email.kntu.ac.ir, khasteh@kntu.ac.ir
[2] Faculty of Electrical Engineering, K. N. Toosi University of Technology, Tehran, Iran
a.nikoofard@kntu.ac.ir
[3] Faculty of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran
shirmohammadi@sru.ac.ir

**Abstract :**
In the game theory, the well-known solution to obtain the best profit in non-repeated games as much as possible is the Nash equilibrium. However, in some repeated non-cooperative games, agents can achieve more profit than the Nash equilibrium by tacit collusion. One of the methods to achieve profit more than Nash equilibriums in tacit collusion is reinforcement learning. However, reinforcement learning-based methods consider only one step in the learning process. To achieve and improve profit in these games, more than one step can be used. In this regard, a learning-based forwarding N-steps algorithm called Forwarding Steps (ForSts) is proposed in this paper. The main idea behind ForSts is to improve the performance of agents in non-cooperative games by observing the last N-step rewards. As ForSts is used in the game theory to learn tacit collusion, it is evaluated by the iterated prisoner's dilemma and the Cournot market. Prisoner's Dilemma is an example of a traditional game. The results show that in the iterated prisoner's dilemma, the agents using ForSts achieve better profit than the agents playing in the Nash equilibrium. Also, in the Cournot electricity market, sum of the profit of agents using ForSts is 3.614% more than the sum of profit of agents` playing in the Nash equilibrium.
**Keywords:** Cournot, Electricity market, Nash equilibrium, Non-cooperative repeated games, Prisoner's Dilemma, Reinforcement learning.

## 1. Introduction

Game theory can be used to solve problems in different fields such as electricity markets, economics, psychology, and computer science. The main participants in games to solve the problems in these fields are agents. The main desire of agents is to achieve more profit as much as possible. However, cooperation between these agents can provide more profit. Based on the cooperation between agents, games can be categorized into non-cooperative and cooperative ones. As their names imply, in cooperative games, agents try to maximize their joint profit while in non-cooperative games the aim is each agent's profit.

However, most of the problems in the industry can be solved by non-cooperative games. In these games, agents try to maximize their profit in one step and the well-known solution to obtain the best profit is Nash equilibrium. Each agent is assumed to know the equilibrium strategies of the other agents, and no agent can gain more profit than Nash equilibrium only by changing his strategy [1]. It is proved that all non-cooperative games have at least one Nash equilibrium [1].

It is shown that in non-cooperative repeated games, agents can achieve more profit by the use of learning methods [3][4]. They can learn to trust each other and gain more profit than the Nash Equilibrium. In other words, tacit collusion can occur when more than one agent changes his strategy. One of the tacit collusion learning

[1] Submission date:09, 03, 2020
Acceptance date: 05, 12, 2020
Corresponding author: Zahra Shirmohammadi, Department Faculty of Electrical Engineering, K. N. Toosi University of Technology, Tehran, Iran

approaches is reinforcement learning [2]. In reinforcement learning, the agent improves its policy by interacting with the environment [2,25]. At each step, the agent takes action and receives a reward from the environment. Reinforcement learning can be used to achieve more profit in repeated game cases.

Several studies in the literature have been done on non-cooperative repeated games to improve the profit. Some of these works investigate the influence of cooperation on the amount of collusion [3]–[5]. Based on these studies, cooperation is important for collusion results, especially for more than two agents. Most of the learning approaches used for teaching agents in games use reinforcement learning. Some reinforcement learning methods in games are applied for oligopoly models, especially Cournot Oligopolies [6]–[10]. In these models, it is proved that some factors are important to get a coherent result. This result depends on the specific details learning process. In Ref [7] Q-learning model is used in Oligopoly Cournot with two levels of production. Computer simulations illustrate that this kind of learning leads to collaboration if the agent does not have a memory, except if machine strategies cannot be used and even if there are more than two agents. Q-learning is a model-free reinforcement learning technique that works by learning the expected utility of action-value pairs for a given action in a given state following the optimal policy thereafter. When the utility is learned, the optimal policy can be constructed simply by selecting the action with the highest value in each state. Q-learning can handle problems with stochastic transitions and rewards. It has been proven that Q-learning will eventually find the optimal policy, in terms of the expected value for the total reward [11]. All of these methods consider only one step in the learning process. Some of the papers do not consider multi-agent[26].

Although in repeated non-cooperative games, agents try to maximize their profit in all games considering Nash equilibrium, the problem is that agents only try to maximize their profit in this one step and they don`t consider profit that they can achieve in future steps of the games. Because of this behavior, all agents try to change their behavior and to learn the behavior of other agents. As these games are a real-world problem, achieving more profit can help agents to make better decisions and get better results. In non-zero-sum games, the sum of benefits for agents is not zero. In these games, althougth the agents have differences of interest, they can cooperate. In non-repeated games, for two agents that do not have contract and cooperation, the maximum benefit is Nash equilibrium. However, in repeated games, the sum of benefits is not zero and this can be used as a motivation to use learning methods. In other words agents without having a contract can reach more benefits than Nash equilibrium by the use of rewards in learning methods.

To solve the above-mentioned problem in repeated non-cooperative games and to learn tacit collusion and to achieve more profit; this paper provides a novel reinforcement learning algorithm called forwarding N-steps reinforcement (ForSts). As a result, tacit collusion agents achieve more profit. Agents who are using ForSts observe next n steps rewards because when the agents observe next n steps rewards, the problem of thinking about just this step will be solved and they care about future rewards. in the case that agents learned successfully, they can make tacit collusion. Results show that agents who use ForSts achieved 300% more profit than the Nash equilibrium in prisoner's dilemma with respect to Q-learning. This idea is tested by Prisoner's Dilemma problem and Cournot equilibria. In the Cournot electricity market sum of profit of agents who use ForSts is 3.614% more than the sum of agents in Nash equilibrium with respect to Q-learning. Also, to analyze the impact of

uncertainties of each parameter of the market in the final results, sensitivity analysis is performed and shows that in the real market, even though with uncertain parameters, tacit collusion occurs and provides more profits for generators in the electricity markets.

The rest of this paper is structured as follows: related case studies are presented in Section 2. Then, in Section 3, the proposed algorithm ForSts is described. In Section 4 simulation results are discussed and eventually the conclusion is presented in Section 5.

## 2. Related Case Studies

Reinforcement Learning (RL) uses reward functions to learn effective behavior between a variety of tasks and environments in some cases without any environmental reward [27]. In some previous works, Multi-Agent Reinforcement Learning (MARL) (e.g.[28]-[30]) is considered. These works focus on value function decomposition, consensus, and learning to communicate between agents [31]. Here, we consider the problem of repeated non-cooperative games and achieve more profit by using N-steps reinforcement learning in the repeated non-cooperative games such as Iterated Prisoner's dilemma and electricity market.

Concerning the fact that Prisoner's dilemma and Cournot are two well-known case studies in game theory; in this section, they are discussed in more detail.

The Prisoner's dilemma is a two-agent non-zero-sum game. In this game, there are two actions; cooperate and defect. Each agent may receive one of the four possible outcomes. Reward (R) outcome, which is obtained by both agents if they both cooperate, punishment (P) outcome which is obtained by both agents if they both defect, temptation (T) outcome which is obtained by the agent who is defecting against the cooperating agent and sucker (S) outcome, which is obtained by the agent who cooperates with the defecting rival. In the prisoner's dilemma, we have $T > R > P > S$ and $2R > T + S$. The prisoner's dilemma is a one-stage game. If it is played repeatedly, it will be called the Iterated Prisoner's Dilemma (IPD). The Prisoner's dilemma has been studied for more than 50 years [12]. In a prisoner's dilemma defect is a dominant strategy.

One of the early studies in the Prisoner's dilemma has been done by Flood [13]. Flood applied this game with the help of two economists. Theoretically, it is predicted that both agents always defect in the Prisoner's dilemma since it is a dominant strategy, but practically, Flood observed cooperation. Cooperation in the Prisoner's dilemma is proposed in [14]. This paper indicates that incomplete information about the other agent can be the reason for cooperation. Due to this reason, each rational agent wants to cooperate. Cooperation can only be achieved before the last step. In [15], the Iterated Prisoner's Dilemma was studied experimentally. These experiments were used to understand the factors affecting an agent's behavior. They indicated that the agents cooperate conditionally. Another approach to solving the IPD problem is learning in games. In [16], a reinforcement learning method has been proposed to learn the best strategy for an agent to achieve more profit. The "Social Reward Shaping" is used to accelerate the learning process. The method is tested against the Tit-For-Tat, always-C, and Q-learning strategies. Based on the game results, the agent using the Social Reward Shaping achieves good rewards against all agents who have used different strategies. Sometimes the agent converged to defect strategy. Since Q learning observes immediate reward and defecting is a dominant strategy, the agent using social reward shaping can partially compensate for the consequences of not observing the next n-step rewards. Inspired by the solutions to the Prisoner's Dilemma problem, researches have been done to solve more complex social dilemma

problems. In [17], a method is proposed for learning, in the fruit gathering and woolpack hunting games, based on the learning methods used to solve the prisoner's dilemma problem. The method uses deep Q-networks for each agent. In [18], a learning method is proposed for an agent trying to cooperate in social dilemmas. It is shown that using the reinforcement learning methods Tit-For-Tat strategy for Prisoner's Dilemma problemis is generalized. The researchers created agents that cooperated first, trying to avoid being exploited, and returned to mutual cooperation. They showed theoretically and experimentally that their agents can reach cooperation in complex games. This method is tested on a coin game.

In a Cournot competition, each firm decides on the amount of its products based on its expectations from the other firms. Thus, if an equilibrium is reached, each firm's expectations of how the other firm's act, is correct, and no firm have the desire to change its output decision [19]. This equilibrium is called "the Cournot-Nash equilibria" [20].

In the Cournot equilibria, if all the sellers determine the output that they produce, none of them have the desire to change its output, as for any seller changing its output; the profit of this agent is reduced. This equilibrium is often called "Cournot-Nash Equilibrium". In this equilibrium, the sellers achieve less profit than the maximum possible profit, because the sellers don't trust each other. If the sellers cooperate to make a cartel, they can reduce their productions, therefore, the price will increase, hence they will achieve more profit in this situation. If one of the seller's defects, he will gain more profit while the other agents will gain less. In [21] an analytic approach is proposed to evaluate the potential of collusion in the electricity market. Their method allows the market regulator to assess the level of competition in the electricity market. In [22], the model of collusion is presented in the form of a mathematical program. Ref [22] used the Power pool market as a game. In [20], a method called SA-Q-learning is proposed which is a modified version of Q-learning to learn the best strategy in the electricity market. Results of [20] demonstrate the development of Tacit Collusion among generators. In [23], both cooperative and non-cooperative supply chains are modeled. In both models, once, the buyer was considered the leader and then the seller is the follower. It is observed that the retailer price in the cooperative model was less than the one in the non-cooperative model. In [7], the Q-learning is used for the learning agent in the Cournot-Oligopoly market. They used Q-learning with memory and Q-learning without memory, separately. It is observed that Q-learning agents could learn to cooperate. In all of the mentioned works, it is possible to achieve more profits; furthermore, some of these methods couldn't be applied to other game theory problems.

## 3. The Proposed Algorithm The roposed Algorithm: Forwarding N-steps Reinforcement Learning (ForSts)

In this section, to achieve more profit than Nash equilibrium in non-cooperative repeated games, we propose an algorithm called Forwarding N-steps Reinforcement Learning (ForSts). The main novelty in ForSts concerning other state-of-the-art methods is that each agent decides based on the next n-step rewards. We use a reinforcement learning approach, to let the agents learn the nearly optimal strategy. This approach is like Q-learning but instead of observing only the immediate step; it observes the next n-step rewards and updates its Q Table by a weighted average of these rewards. In this approach, the

applied reward is a weighted average of the next n-step rewards and the weights determine the strategy which is more valuable for the agent to learn a good policy.

One of the important parameters of ForSts is the number of steps the learners observe. This parameter affects the learning process; if this number is too small, the learners won't converge to cooperation, and if it's too large, the learners won't learn a good policy. The agent should maximize its payoff. A learner agent should try to cooperate against the agent who has the incentive to cooperate. In addition, the learner agent defects against the agent which has the incentive to defect. The learner agent must also defect against the agent which always cooperates or defects without considering the other agent.

Another important parameter of ForSts is the exploration rate that affects the number of visited state-action pairs. It is initialized to a predefined value in the first step of the game and decreases linearly until it reaches zero.

The action selection mechanism is ε-greedy. In this mechanism, in each state, the action which has the maximum Q-value is selected with a probability of 1- ε, and a random action is selected with a probability of ε (exploration rate).

ForSts can be used to learn the nearly optimal policy in different game theory problems. We can use arbitrary value as the initial value of the Q-Table, but if we choose the initial values intelligently according to the problem which needs to be solved, the convergence speed is improved. The pseudo-code of ForSts is presented in Fig. 1.

Learning rate determines how much we care about new rewards instead of the previous Q-value. The number of steps each agent observes is n. Weights are the coefficients of rewards of the next n-steps. We update the Q-value of a state-action pair after observing the next n-state action pairs because we need the values of the next n rewards to update the Q-value. In the beginning, when we are at the first n iterations, we only use the immediate reward to update the Q-values. After n iterations, we began to update the Q-value of the state which is the n step before the current state using all rewards of the last n-steps. The flowchart of ForSts is shown in Fig. 2.

1. Initialize Q table
2. Initialize discount factor, initial exploration rate, learning rate, number of observing states(n), weights (the coefficients of rewards of the next n-steps,$w_1, w_2, \ldots, w_n$), and number of steps
3. For $i = 1$ to number_of_steps do
    a. Select an action based on the ε-greedy mechanism and get its immediate reward ($r_i$)
    b. If $i > n$ then
        i. $R = 0$
        ii. For $j = 1$ to n do

$R = R + r_{i+1-j} * w_j$ (R is the applied reward, which is the weighted average of rewards of the n-steps)

$Q(s_{i+1-n}, a_{i+1-n})$
$= Q(s_{i+1-n}, a_{i+1-n}) + learning\_rate$
$* (R + discount\_factor$
$* \max_a(Q(s_{i+1}, a_{i+1})) - Q(s_{i+1-n}, a_{i+1-n}))$

    c. else

$R = r_i$, (R is the applied reward)

$Q(s_i, a_i)$
$= Q(s_i, a_i) + learning\_rate * (R$
$+ discount\_factor$
$* \max_a(Q(s_{i+1}, a_{i+1})) - Q(s_i, a_i))$

**Fig. 1. Pseudo code**

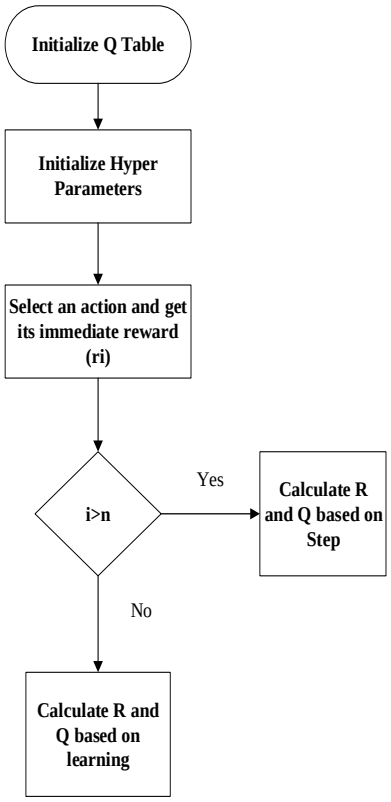


**Fig 2. Flowchart of ForSts**

## 4. Simulation results

In this section, the simulation results are presented to investigate the effectiveness of ForSts. ForSts is tested to learn a nearly optimal policy in the iterated prisoner's dilemma and the Cournot's duopoly market. Results are compared with the Q-learning method which is used to solve the same problems. The Q-learning algorithm presented in [16] is used, but due to the difference in the values of rewards, the presented results are obtained from implementations. The result achieved from the implemented algorithm is the same as [16] in the condition that the values of rewards are the same with the values of the ref [16].

### 4.1. Iterated Prisoner's Dilemma

In the iterated prisoner's dilemma, the learner agent is tested once against itself, and once against the agent using the Tit-For-Tat, always-C, and always-D strategies. The outcome of R (cooperate-cooperate) is 4, the outcome of T (when our agent defects and the other cooperates) is 5, the outcome of P (defect-defect) is 1 and the outcome of S (when our agent cooperates and the other defects) is 0. The game is played for 10000 steps. In the experiments, the initial value for all state-action pairs is 0.5. The initial exploration rate was 0.6 in the first step and decreased linearly until it reached zero in step 8000. The discount factor was 0.9. The agent saw the next 3 steps. The weight of the first step was 0, the weight of the second step was 0.3 and the weight of the third step was 0.4. Many different values for these parameters were tested and achieved better results with the mentioned values.

Fig. 3 indicates the reward which has been achieved by our learning agent in each iteration. In this simulation, the vertical line is step 8000. After step 8000, the exploration rate is set to 0.

Our agent has played against itself and the agents using the other strategies such as the Tit-For-Tat, always-C, and etc. It is observed that the results for our agent, compared to the Q-learner agent, are more favorable.

The presented results are the average of 20 times stimulation running. Furthermore, each presented reward is an average of the achieved rewards of the last 100 steps; we do that to smooth the curve.

In Fig. 3, an agent using the ForSts algorithm plays against an agent who always cooperates. In this situation, we expect the learner to choose a defect strategy (the outcome T). The best action against an agent using the always-C strategy is defecting all the time because by defecting the best reward will be achieved. The results are compared to a Q-learner agent; both agents converge to defecting after step 8000. This indicates both algorithms behave efficiently against the

always-C strategy.

In Fig. 4, an agent using the presented algorithm plays against an agent who always defects. Similar to always-C, the best action against an agent using the always-D strategy is defecting all the time because by defecting the best reward will be achie ved. The results are compared to a Q-learner agent; both agents converge to defecting after step 8000. This indicates both algorithms behave efficiently against the always-D strategy.

In Fig. 5, an agent using the ForSts algorithm plays against an agent using the Tit-For-Tat strategy. Unlike previous strategies, the best action against the agent using the Tit-For-Tat strategy is cooperating all the time. The results are compared to a Q-learner agent; both agents converge to cooperating after step 8000. These indicate both algorithms behave properly against the Tit-For-Tat strategy.

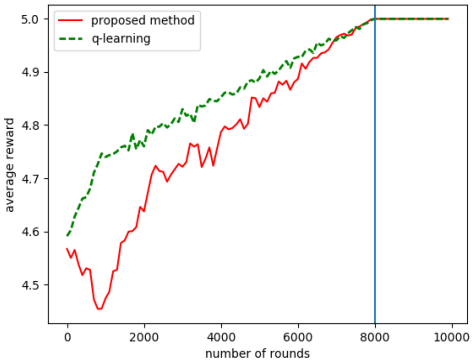


**Fig. 3. Comparison of the average achieved reward against the always-C strategy.**

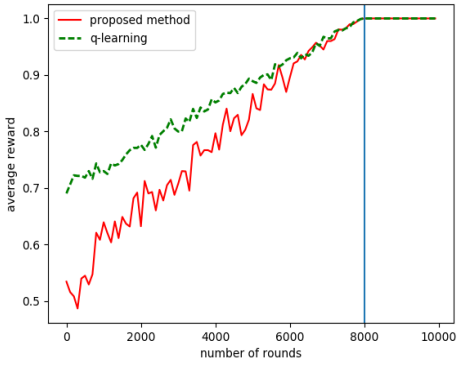


**Fig. 4. Comparison of the average achieved reward against the always-D strategy.**

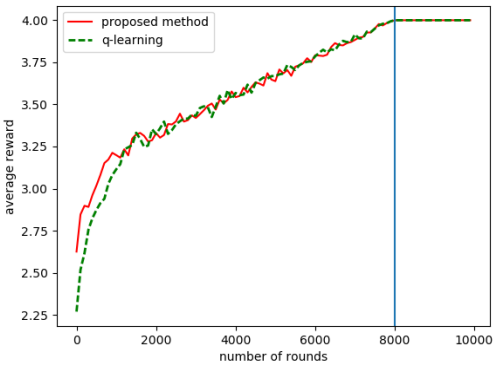


**Fig. 5. Comparison of the average achieved reward against the Tit-For-Tat strategy.**

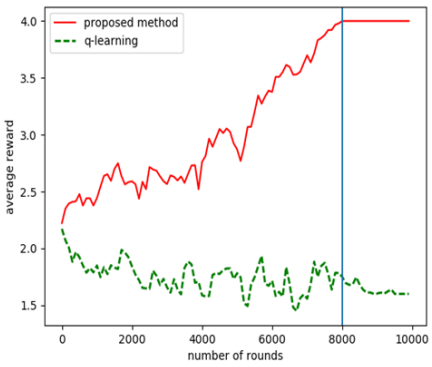


**Fig. 6. Comparison of the average achieved reward against the agent using the same strategy.**

In Fig. 6, an agent using the ForSts algorithm plays against an agent using the same strategy. This is the most important test for a learning agent. The best outcome in this situation is expected to be cooperate-cooperate (the outcome of R). Based on these results it is shown that the ForSts algorithm converges to cooperate-cooperate after 8000 steps but Q-learner agents cannot converge to the best outcome. The average achieved reward for the last 100 steps is considered. This process runs for 2000 times and the average of these 2000 runs is reported in Table 1.

**Table (1): The profits of our learning agent in different cases**

| Type | Profit |
|---|---|
| Learner | 3.977 |
| Tit-For-Tat | 4 |
| Always-C | 5 |
| Always-D | 1 |

The best possible result when the learner agent is playing against itself is converging to cooperate-cooperate and achieving an outcome of 4. Based on the results presented in Table 1, the average achieved reward by our learning agent is 3.977 which is very close to 4, hence in 98.7%of the iterations, the two agents have converged to cooperate-cooperate. The best possible strategy against an agent using the Tit-For-Tat strategy is to always cooperate and achieve the outcome of 4. The results show that the agent has done perfectly well and it has achieved the outcome of 4. The best possible strategy against an agent that is using the always-C strategy is to always defect and achieve the outcome of 5. The results show that the agent has converged to this strategy and achieved the outcome of 5 every time. The best possible strategy against an agent that is using the always-D strategy is to always defect and achieve the outcome of 1. We can see that the agent has always converged to the best possible strategy (always defecting).

## 4.2. Cournot's duopoly model

Also, ForSts on Cournot's duopoly model is applied. In this model, we consider two agents and each agent has a specific cost function. All agents produce the same product. *q1* is the quantity of the production for the seller 1 and *q2* is the quantity of the production for the seller 2. *cost1* and *cost2* are the costs for the seller 1 and the seller 2, which are shown in equation 1 and equation 2, respectively. In these equations *a1*, *a2*, *b1*, *b2*, *c1*, and *c2* are the coefficients of cost functions.

$$cost_1 = a_1 * q_1 * q_1 + b_1 * q_1 + c_1 \quad (1)$$

$$cost_2 = a_2 * q_2 * q_2 + b_2 * q_2 + c_2 \quad (2)$$

The parameter values of the cost function of the agents which have been used in simulations are given in Table 2.

**Table (2): The parameter values of the cost function of the agents**

| Parameters | Value | Parameters | Value |
|---|---|---|---|
| $a_1$ | 0.2 | $a_2$ | 0.17 |
| $b_1$ | 1.8 | $b_2$ | 2 |
| $c_1$ | 70 | $c_2$ | 50 |

The profit of the agents and the price of the market are given by equation 3. *Profit1* and *profit2* are the profit for the seller 1 and the seller 2, that can be calculated by, equation 4 and equation 5, respectively

$$price = 20 - 0.18 * q_1 - 0.16 * q_2 \quad (3)$$
$$profit_1 = q_1 * price - cost_1 \quad (4)$$
$$profit_2 = q_2 * price - cost_2 \quad (5)$$

The values for cost function, profit and the price parameters, also profit and cost function equations of the agents are taken from [24]. The Nash equilibrium for this game indicates that the production quantity for seller 1 is 21 and the production quantity for seller 2 is 20.6. Under these circumstances, seller 1 will achieve a profit of 106.48, and seller 2 will achieve a profit of 73.87. To test ForSts, this game is played for 10000 steps. The initial Q-values for all state-action pairs were 1000. The initial exploration rate was 0.6 in the first step and decreased linearly until it reached zero in step 8000. The discount factor was 0.9. The agent observes the next 3 steps. The weight of the first step was 0, the weight of the second step was 0.4, and the weight of the third step was 0.3. We tested many different values for Both agents in the simulation used the same learning method. Fig. 7, Fig. 8, and Fig. 9 show the rewards achieved by seller 1, seller 2, and the summation of rewards for seller 1 and seller 2. The presented results are a comparison between usages of ForSts versus the Q-learning method. The vertical line is a step. After step 8000, the exploration rate is set to 0.

The horizontal line is the reward achieved by the agents using the Nash equilibrium strategy.

The presented results are the average of 20 times stimulation running. Furthermore, each presented reward is an average of the achieved rewards of the last 100 steps; we do that to smooth the curve. Fig. 7, Fig. 8, and Fig. 9 indicate that the learner agent can cooperate and achieve more profit than the Nash equilibrium. Results show that agents using our learning method cooperate more than the agents using the Q-learning method. We use the Q-learning algorithm which is presented in [20], but due to the difference in the purpose of learning, the presented results are obtained from implementations. Table 3 compares the average rewards of the last 1000 steps of the game, which is considered as the output of two methods.

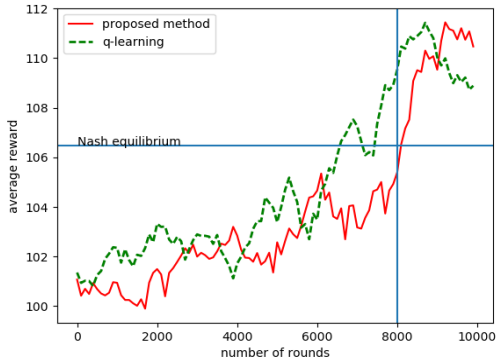


**Fig. 7. Comparison of the average achieved reward for the seller 1**

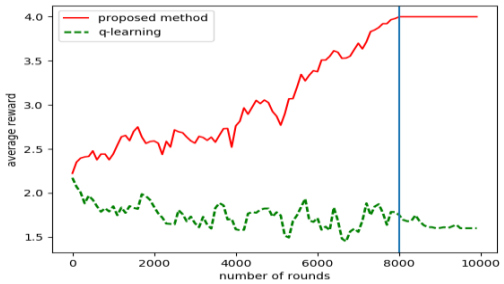


**Fig. 8. Comparison of the average achieved reward against the agent using the same strategy.**

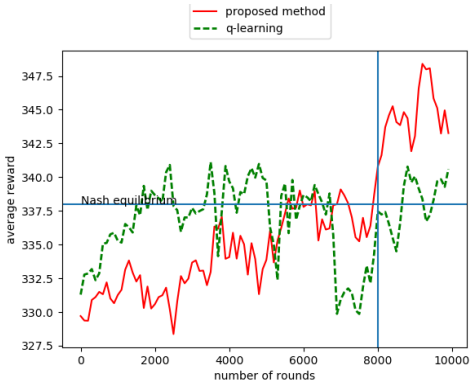


**Fig. 9. Comparison of the average achieved reward for the player 1**

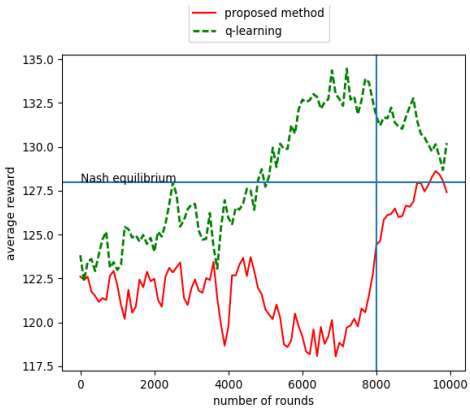


**Fig. 10. Comparison of the average achieved reward for the player 2**

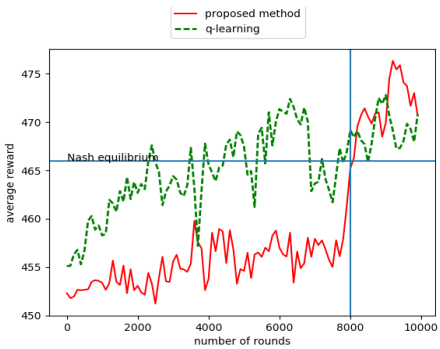


**Fig. 11. Comparison of the average achieved reward for sum of palyers**

Besides, to test the results with other cost functions and prices, another case study is simulated. In this simulation, Costs of seller 1 and seller 2 are shown as *cost1* and *cost2 i*n equation 6 and equation 7, respectively. In equation 8 the price of the market is also given.

$$cost_1 = q_1 * q_1 \quad (6)$$

$$cost_2 = 15 \times q_{2+}\, q_2 \times q_2 \quad (7)$$

$$price = 60 - q_1 - q_2 \quad (8)$$

The Nash equilibrium in this game for player 1 is 13 and for player 2 is 8.

Based on our simulation results a Nash profit of 338 and 128 is achieved by player 1 and 2, respectively. These results are played 10000 steps to test ForSts. Q-values for all state-action pairs are considered 1000. In the first step, the initial exploration rate was 0.6 and decreased linearly until it reached zero in step 8000. The discount factor, has the same scenario as our previous case study. Both players in the simulation used the same learning method. Fig. 10, Fig. 11, and Fig. 12 show the rewards achieved by player 1, player 2, and the summation of rewards for player 1 and player 2, respectively. Table 4 compares the average rewards of the last 1000 steps of the game, which is considered as the output of two methods.

**Table (3): Comparison of the average reward of last 1000 steps of the game**

| Method | Seller 1 | Seller 2 | Summation |
|---|---|---|---|
| ForSts | 110.81 | 187.61 | 76.80 |
| Q-learning | 109.32 | 185.97 | 76.64 |

**Table (4): Comparison of the average rewards of last 1000 steps of the game**

| Method | Player 1 | Player 1 | Summation |
|---|---|---|---|
| ForSts | 345.1 | 128.1 | 473.2 |
| Q-learning | 339.8 | 129.8 | 469.6 |

**Table (5): Sensitivity analysis**

| | Without change | $a_1$+10% | $a_1$-10% | $b_1$+10% | $b_1$-10% | $a_2$+10% | $a_2$-10% | $b_2$+10% | $b_2$-10% |
|---|---|---|---|---|---|---|---|---|---|
| Agent 1 : Profit of learning | 111.114 | 105.614 | 112.272 | 107.484 | 112.776 | 107.799 | 107.357 | 106.866 | 109.963 |
| Agent 1: profit of Nash equilibrium | 106.485 | 98.402 | 115.496 | 102.032 | 111.003 | 110.271 | 102.279 | 107.383 | 105.589 |
| Agent 2: Profit of learning | 75.769 | 77.224 | 75.239 | 76.926 | 76.986 | 67.477 | 86.780 | 77.396 | 75.504 |
| Agent 2: profit of Nash equilibrium | 73.878 | 77.630 | 69.775 | 74.971 | 72.790 | 65.440 | 83.434 | 70.087 | 77.720 |
| Sum of both agent: Profit of learning | 186.884 | 182.839 | 187.512 | 184.410 | 189.762 | 175.276 | 194.138 | 184.263 | 185.468 |
| Sum of both agent: profit of Nash equilibrium | 180.364 | 176.033 | 185.272 | 177.003 | 183.793 | 175.712 | 185.713 | 177.471 | 183.310 |

## 4.3. Sensitivity Analysis

Usually, the parameters of the market have uncertainties. To analyze the impact of uncertainties of each parameter of the market in the final results, sensitivity analysis is performed. The final result is calculated by using equations 1 and 2 (cost functions). This test is performed to analyze the impact of uncertainties of the values of the parameters of the cost functions (a1, b1, a2, b2).

10% uncertainty in each parameter is

assumed. We analyze the effect of ±10% change in the value of each parameter, where the other parameters are fixed. Each game is played 10 times and the results are averaged. The presented profit is the average profit of the last 100 steps of the learning agents. The results are as shown in Table 5.

The results show that the tacit collusion occurs in the presence of uncertainties in the parameters of the cost function of the other agent except for some cases and this is promising because cooperation occurs in almost all cases. When a parameter of the cost function decreases, the cost will decrease and consequently, the net profit will increase, but when a parameter increases, the net profit will be decreased.

In the case in which $b_2$ increases by 10%, the difference between the profits of Nash equilibrium and learner agent is almost equal to the case in which the values of the parameters have not changed, so the results show that this method is not sensitive to the increase of $b_2$. In the case in which $a_2$ increases by 10% the difference between the profits of the Nash equilibrium and the learner agent is not equal to the case in which the parameters of the cost function of agent 1 have not changed, so these results show that the algorithm is sensitive to the increase of $a_2$.

## 5. Conclusions

In this paper, a novel reinforcement learning algorithm called ForSts is proposed for the agents in the non-cooperative games to achieve more profit than the Nash equilibrium. In ForSts, the agent observes the next n-step rewards. The results in the Iterated Prisoner's dilemma show that the agent using the approach had learned the nearly optimal strategy. The agent achieved the best possible results against the agents using the always-D, always-C, and the Tit-For-Tat strategies; furthermore, the agent almost achieved the best results possible against itself. Also, the results show that the agents in Cournot's duopoly using the proposed method had learned to achieve more profit than the Cournot-Nash equilibrium.

## Acknowledgment

This work was supported by Shahid Rajaee Teacher Training University.

### References

[1] M. J. Osborne and A. Rubinstein, A course in game theory. MIT Press, 1994.
[2] R. S. Sutton and A. G. Barto, "Introduction to reinforcement learning (Vol. 135)." Cambridge: MIT Press, 1998.
[3] C. Engel, "Tacit collusion: The neglected experimental evidence," J. Empir. Leg. Stud., vol. 12, no. 3, pp. 537–577, 2015.
[4] M. A. Haan, L. Schoonbeek, and B. M. Winkel, "Experimental results on collusion," Exp. Compet. policy, vol. 9, 2009.
[5] J. Potters and S. Suetens, "Oligopoly experiments in the current millennium," J. Econ. Surv., vol. 27, no. 3, pp. 439–460, 2013.
[6] S. S. Izquierdo and L. R. Izquierdo, "The 'Win-Continue, Lose-Reverse' rule in Cournot oligopolies: robustness of collusive outcomes," in Advances in Artificial Economics, Springer, 2015, pp. 33–44.
[7] L. Waltman and U. Kaymak, "Q-learning agents in a Cournot oligopoly model," J. Econ. Dyn. Control, vol. 32, no. 10, pp. 3275–3293, 2008.
[8] S. O. Kimbrough and M. Lu, "A note on Q-learning in the Cournot game," in Proceedings of the second workshop on e-business, 2003.
[9] S. O. Kimbrough, M. Lu, and F. Murphy, "Learning and tacit collusion by artificial agents in Cournot duopoly games," in Formal modelling in electronic commerce, Springer, 2005, pp. 477–492.
[10] G. Tesauro and J. O. Kephart, "Pricing in agent economies using multi-agent Q-learning," Auton. Agent. Multi. Agent. Syst., vol. 5, no. 3, pp. 289–304, 2002.
[11] F. S. Melo, "Convergence of Q-learning: A simple proof," Inst. Syst. Robot. Tech. Rep, pp. 1–4, 2001.
[12] T. W. Sandholm and R. H. Crites, "Multiagent reinforcement learning in the iterated prisoner's dilemma," Biosystems, vol. 37, no. 1–2, pp. 147–166, 1996.
[13] M. M. Flood, "Some experimental games," Manage. Sci., vol. 5, no. 1, pp. 5–26, 1958.
[14] D. M. Kreps, P. Milgrom, J. Roberts, and R. Wilson, "Rational cooperation in the finitely repeated prisoners' dilemma," J. Econ. Theory, vol. 27, no. 2, pp. 245–252, 1982.
[15] M. Embrey, G. R. Fréchette, and S. Yuksel, "Cooperation in the finitely repeated prisoner's dilemma," Q. J. Econ., vol. 133, no. 1, pp. 509–551, 2017.
[16] M. Babes, E. M. De Cote, and M. L. Littman,

"Social reward shaping in the prisoner's dilemma," in Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3, 2008, pp. 1389–1392.

[17] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, 2017, pp. 464–473.

[18] A. Peysakhovich and A. Lerer, "Maintaining cooperation in complex social dilemmas using deep reinforcement learning," 2018.

[19] H. R. Varian, Intermediate Microeconomics: A Modern Approach: Ninth International Student Edition. WW Norton & Company, 2014.

[20] A. C. Tellidou and A. G. Bakirtzis, "Agent-based analysis of capacity withholding and tacit collusion in electricity markets," IEEE Trans. Power Syst., vol. 22, no. 4, pp. 1735–1742, 2007.

[21] M. Samadi and M. E. Hajiabadi, "Assessment of the collusion possibility and profitability in the electricity market: A new analytical approach," Int. J. Electr. Power Energy Syst., vol. 112, pp. 381–392, 2019.

[22] S. Jahanbakhshi, H. Khaloozadeh, and A. Nikoofard, "Tacit collusion in pool-based electricity markets with a demand shock," in Electrical Engineering (ICEE), Iranian Conference on, 2018, pp. 1197–1202.

[23] M. Esmaeili, M.-B. Aryanezhad, and P. Zeephongsekul, "A game theory approach in seller-buyer supply chain," Eur. J. Oper. Res., vol. 195, no. 2, pp. 442–448, 2009.

[24] H. Kamalinejad, V. J. Majd, H. Kebriaei, and A. Rahimi-Kian, "Cournot games with linear regression expectations in oligopolistic markets," Math. Comput. Simul., vol. 80, no. 9, pp. 1874–1885, 2010.

[25] N. Jaques, Angeliki Lazaridou, E. Hughes, Caglar Gulcehre, P. Ortega, D. Strouse, Joel Z. Leibo, and N. De Freitas, "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," In Proceedings of the 36th International Conference on Machine Learning, vol. 97 of Proceedings of Machine Learning Research, p.p. 3040–3049, 2019.

[26] A. D. Edwards, L. Downs, and J. C. Davidson. Forward-backward reinforcement learning. ArXiv, 1803.10227, 2018.

[27] S. P. Singh, A. G.Barto, and N. Chentanez, "Intrinsically motivated reinforcement learning. In Advances in Neural Information Processing Systems 17 Neural Information Processing Systems," NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada], pp. 1281–1288, 2004.

[28] H. Mao, Zhengchao Zhang, Z. Xiao, and Z. Gong. "Modelling the dynamic joint policy of teammates with attention multi-agent ddpg," 18th International Conference on Autonomous Agents and Multiagent Systems, pp. 1108–1116. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[29] W. J. K. David Earl Hostallero Kyunghwan Son, D. K., and Y. Yi. " Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," 31st International Conference on Machine Learning, 2019.

[30] K. Zhang, Z. Yang, and T. Basar. "Networked multi-agent reinforcement learning in continuous spaces," IEEE Conference on Decision and Control (CDC), p.p 2771–2776, 2018.

[31] A. OroojJadid, and D. Hajinezhad, 2019. " A review of cooperative multi-agent deep reinforcement learning," arXiv preprint arXiv:1908.03963.