JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

DEVELOPMENT OF AN OPTIMAL ROAD-TRAFFIC CONTROLLER THROUGH REINFORCEMENT LEARNING

Rudolph Joshua Candare, Adrian Pete Lagare, Elly John Mirasol, JaymerJayoma and RolynDaguil

[1]Department of Computer Science, College of Computing and Information Sciences, Caraga State University
Ampayon, Butuan City, Agusan del Norte, Philippines.

[2]Graduate School, Caraga State University
Ampayon, Butuan City, Agusan del Norte, Philippines.

Corresponding Author's Email:[1]rucandare@carsu.edu.ph

Article History: Received xxxxx; Revised xxxx; Accepted xxxx

ABSTRACT: The common way to manage road intersection traffic is done through traffic lights. Existing traffic light intersection controllers have a fixed timing in their outputs without considering the real-time traffic situation. This study aims to develop a smart traffic light controller which is adaptive depending on the traffic situation in order to lessen the traffic in intersections. In order to achieve the smart traffic light controller, reinforcement learning is the technique in which the researchers implement for its development. Reinforcement Learning (RL) is a type of machine learning technique that best fits this problem. It learns by interacting through its environment and given rewards and punishment based on its actions. In this paper, the researchers used two RL methods, namely Deep Q-Learning/Network (DQN) and Covariance Matrix Adaptation Evolutionary-Strategies (CMA-ES) with the same deep neural network architecture to implement the smart traffic light controller and will be simulated via Unity 3D. The smart traffic light agent is tested in two test areas with easy and hard levels of traffic with DQN, CMA-ES together with the Fixed-Time for its basis. The results of this study demonstrate the advantage of using two different RL methods over a fixed time traffic controller. The researchers found out that CMA-ES had better performance for a smart traffic light controller.

KEYWORDS: *Reinforcement Learning, Traffic Control, Deep Q-Learning, Covariance Matrix Adaptation Evolutionary-Strategies, Deep Neural Network*

1.0 INTRODUCTION

The common way to manage road intersection traffic is done through traffic lights. Existing traffic light intersection controllers have a fixed timing in their outputs without considering the

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

real-time traffic situation [1]. This current way of controlling traffic lights is an inefficient way to implement it in intersections as it causes major traffic congestion. It also causes several problems of commuters such as delay going to their destination, road rage, fuel consumption, waste of energy, and the worst, vehicular accidents. If the area still uses the existing fixed, controlled traffic lights, traffic congestion will still continue to get worse due to the infrastructures being constructed from time to time together with more demands for vehicles. This motivates us to develop a smart traffic light controller, which is adaptive that dynamically adjusts the traffic light timing based on the real-time traffic situations happening in an intersection.

In order to achieve the smart traffic light controller, reinforcement learning is the technique in which the researchers implement for its development. Reinforcement Learning (RL) is a type of machine learning technique that best fits this problem. It learns by interacting through its environment and given rewards and punishment based on its actions [2]. The goal of RL is to maximize its rewards, but in order to achieve this, it should learn the most optimal policy during the process. In this case, the goal is to reduce the waiting time of the vehicles queued up in an intersection resulting in smooth traffic flow within the area. The controller must first take the traffic information such as vehicle number in the queue and waiting time for each road in the intersection where it is done through existing technologies [3]. After taking the inputs, it is processed in a deep neural network, and later it outputs the change of timings of the traffic lights.

With reinforcement learning, the researchers can create a smart traffic light controller that can adapt to any real-time traffic situation happening in an intersection. Reinforcement Learning (RL) is modeled as a Markov Decision Process (MDP), which represents a tuple (S, A, P, R) of an agent's state, action, policy, and reward [4]. In each step, the agent will get a representation of an environment's state $s \in S$ and chooses an action $a \in A$ based on some policy $\pi(a \mid s)$, and later will proceed to the next state according to the transition dynamics P, $s' \sim P(s'|s, a)$. After each episode or step, the agent will receive a reward $r = R(s, a)$ as a consequence of its previous action [5]. This process of choosing an action from the current state, receiving rewards, and moving on to a new state occurs continuously in the environment until it reaches to the end state.

In Q-learning, the single agent will be appropriate enough for its training. Its corresponding q-network's output will be q-values based on its state input. Adjusting of weights of the q-network is done through backpropagation. The Q-table is also needed during the learning process since the q-network will learn from the q-table itself. In CMA-ES, its policy network would output direct actions based on a given state input. Since its a policy iteration-based algorithm, CMA-ES will need multi-agent to be trained to have each agent's own policy network. Adjusting of weights of the policy network is generated from the mean value and covariance of the elite candidates. The average performance of both algorithms, together with the fixed time method, will be compared in order to decide which method is more efficient in optimizing traffic signal timing control for the road. The researchers chose Gmall and Ochoa intersection as the location to be tested with the smart traffic light controller since both intersections are one of the busiest intersections in Butuan City especially with SM Mall will be opening soon along JC-Aquino highway (connecting both intersections) which will make both

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

of the intersections much busier.

2.0 REVIEW OF RELATED LITERATURE

Traffic researchers often use computer simulations to model individual vehicles as distinct dynamic objects to reproduce real-world traffic. Simulation of Urban MObility (SUMO) in a vehicular network is also a popular tool that shows the efficiency of the model in controlling traffic lights and also to simulate a traffic system involving cars, roads, intersections, and traffic lights. [1].Since the early 1990s, reinforcement learning was applied in traffic light signal control to dynamically adjust the timing of traffic lights according to real-time traffic [6]. Also, during the 90s, reinforcement learning techniques were limited to tabular Q learning, and a linear function was normally used to estimate the Q value. However, such partial information could not fully represent the complexity of road traffic systems. Because of this past limitation of the technique in reinforcement learning, recent studies apply reinforcement learning with double Q- Learning to reduce the size of the state space, such as with the number waiting for vehicles and the statistics of traffic flow [7][8][9]. There is a steady growth of interest in applying deep reinforcement-learning algorithms to optimal control problems, where convolutional neural networks are used to approximate optimal Q-values.Many innovations in the field of reinforcement learning have gained traction in the RL community, such as deep Q network [10], but only a few of these ideas were significantly effective in traffic control [1].

In the past, the state representation uses discrete table values like the location of vehicles or the number of cars waiting in the queue. This action-pair matrix requires large memory space, making it infeasible for large state space problems. Deep Q-learning methods, on the other hand, avoid the large Q-table problem by using continuous state representations. In these methods, a deep convolutional network is trained to learn a Q-value function which can be used to predict the value or reward of every state-action pairs. These works vary in the state representation, including handcrafted features like queue length, average delay, and image features [11].

Aside from DQN, another re-emerging variant of RL method is The covariance matrix adaptation evolution strategy (CMA-ES). The fast adaptation of step size and covariance matrix makes the CMA-ES one of the most efficient direct search algorithms for real-valued optimization. CMA-ES was proposed for Reinforcement Learning (RL) for the first time in 2003[12]. The CMA-ES was observed to outperform other evolutionary RL approaches on variants of the pole balancing benchmark in fully and partially observable environments.

3.0 METHODOLOGY

The main objective of this study is to develop a smart traffic light controller with reinforcement learning applying its two methods, namely deep q-learning and covariance matrix adaptation evolutionary strategies, in an attempt to maximize the traffic light agent rewards and lessen the waiting time of the vehicles in intersections. This study will be tested in a vehicle and traffic light simulation. The researchers created two (2) test areas of the simulation, namely the Gmall (Figure 1) and Ochoa (Figure 2) intersection based upon Butuan City. During the testing, the researchers will statistically compare the rewards and the average time with an average of 5 runs in DQN, CMA-ES, and fixed time with an easy and hard level.

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

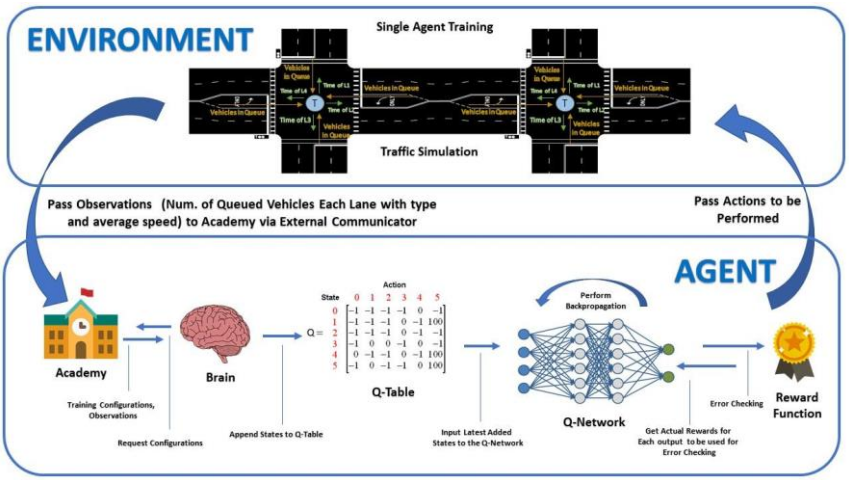

Figure 1. Deep Q-Learning Applied in a Traffic LightController

Figure 1 shows the process of deep q-learning, and Figure 2 for CMA-ES applied for the smart traffic light controller. In this study, there is an agent and an environment. The agent is the decision-maker or the learner.

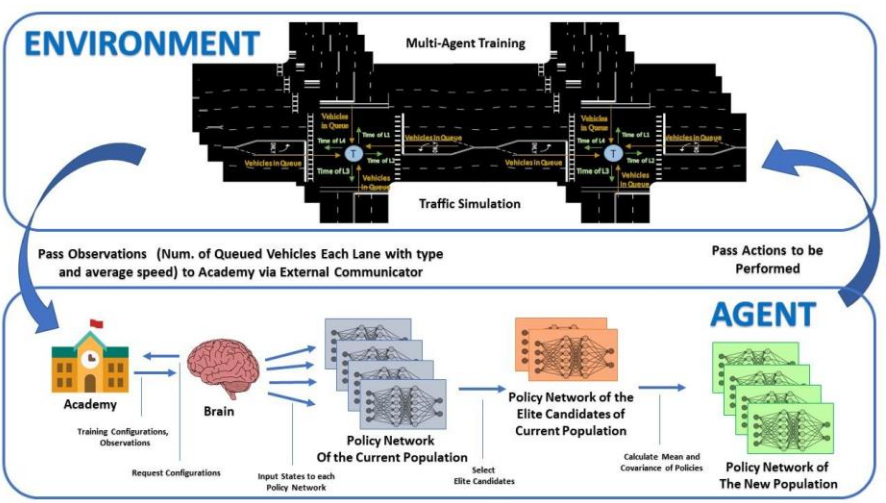

Figure 2.CMA-ES Applied in a Traffic LightController

3.1 Implementing the Simulation

Unity is a development platform where it allows users to create 2D or 3D games. The researchers chose unity because it is easier to work with compared to other game development

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

platform. It also had several assets and plugins available for users. Lastly, unity has a free version, namely Unity Personal. ML-Agents is a plugin in unity which makes unity assets or objects turn into an agent. In this plugin, its environment must be composed of Academy, Brain, and Agent objects with its corresponding scripts. An academy object represents the training environment where the agent's training configurations are defined. It is a school or classroom where the agent will be trained. The brain object is where an agent will base its decision at the same time learning from it. The states, actions, and rewards will be defined in the brain object via script. The agent object is just the physical model of the agent to be trained in an academy and where the brain object will be attached. In this case, the agent is the two traffic lights present in both intersections, and the type of brain the researchers will use is an external brain so that it will be trainable via a neural network.

3.2 Reinforcement Learning Algorithms

In this paper, the researchers used two reinforcement learning methods, namely Deep Q-Learning/Network (DQN) and Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES), to implement the smart traffic light controller. Both algorithm goal is to maximize future rewards but differs on the manner of how they will work and learn. DQN uses value iteration, which uses one policy and improves that policy based on the value improvement of each state in which CMA-ES uses several policies to be tested, then the top-performing policies will be used and will be the basis for next generation of policies to be generated. Deep neural networks with a feed-forward design will be applied to both methods.

3.3 DQN

Deep Q-Learning starts by initializing random values for the parameters or the tuple, state, action, reward, and next state. Next, it initializes the memory with the defined capacity for the experience replay process.

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

DQN Algorithm

1: Initialize parameters, θ with random values
2: Initialize replay memory M with capacity L
3: **for** episode **do**
4: initialize s with current observation of the intersection
5: **repeat step**
choose action α according to €-greedy policy
7: take action α, observe reward r and next state s′
8: store transition (s, a, r, s′) in M
9: s ←s′
10: b ←sample batch of transitions from the replay memory, M
11: **for** each transition (sj , aj , rj , s′j) in b **do**
12: **if** s′j is terminal then
13: yi ←rj
14: **else**
15: yj = rj + γmaxa0 Q(sj , a′ ; θ – i –1)
16: **end if**
17: update parameters θ through backpropagation
18: **end for**
19: **until** s is terminal
20: **end for**

In each episode, it initializes s with a current observation of the intersection. With that state, choose an action according to a greedy policy. With that action, a reward is given and will proceed to the next. A sample batch will be retrieved in the memory. With each transition or possible state in the current state, then update the q-values. It will be repeated until all batch, state, and episodes are done.

3.4 CMA-ES

CMA-ES starts by initializing random neural networks for all the agent population and a fitness function. It then iterates through all generations, episodes, and agents. It each agent, the state is observed. The agent will get a reward based on that observation and will transition to the next state. After the iteration, the population is then sorted in descending order (high rewards first) based on rewards. Next, the elite candidates will be chosen from the current population, and its mean and covariance will be calculated. The children will then be generated based upon sample multivariate normal distribution. The population will then be updated to a newer one and will be returned accordingly.

CMA-ES Algorithm

1: Input: population P of randomly initialized neural networks, fitness function F.
2: **for** g = 1 ,2… , in G generations **do**
3: **for** ep = 1,2… , in Ep episodes **do**
4: **for** agent = 1,2… , in P **do**

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

5: observe state S each step
receive reward R , R = F (S), based on the currentstate S,
7: perform action a = π (S) based on current policy π,
8: Sort P with descending order by reward R
9: Set Elite Candidates C from P
10: Calculate Mean M and Covariance Cov of C
11: **for** i = 1 ,2 … in P-C **do**
12: Children[i]= Sample_Multivarite_Normal(M,Cov)
13: Update New P , P = { C, Children}
14: Return: P
15: **end for**

4.0 EXPERIMENTAL RESULTS AND DISCUSSIONS

4.1 Unity 3D Simulation

The simulation runs successfully with all the setup the researchers prepared. The traffic lights and vehicles are working properly since the vehicle followed the behaviors,the researchers tweaked and also followed the traffic light signals. The red and blue spots present on the road will be gone during the exporting of the simulation. With all of this, the researchers successfully simulated a road and traffic simulation with an intersection, as shown in Figure 3.
Figure 3. Traffic Simulation via Unity 3D

4.2 Architecture and Training Hyperparameters

The network has 2 layers: 4 input units, 128 hidden units, and 8 output units. The network is a feed-forward type with an activation function of tanh. Tanh activation function will rescale the

values of the network nodes to -1 to 1. Input nodes correspond to the observations in each lane. There are four input nodes since there are four lanes to collect observations with. The hidden units are set to 128 nodes since it is a common node count for a small to medium problems. The

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

output units correspond to a certain time the traffic light will give to the available lane to go. The following table shows the hyperparameters that were applied to the agent's brain.

Table1. Configured Hyperparameters set for Training the Policies

HyperParameters	**Q-Network**	**CMA-ES**
Max Episodes	100	100
Population	N/A	10
Elite Candidate Selection Value	N/A	0.2
Evolve Rate	N/A	0.2
Episode Reset Time	300 sec.	300 sec.
Number of Layers	2	2
Input Units	4	4
Hidden Units	128	128
Output Units	8	8

The researchers have gone a few trials and errors during the setting of hyperparameters since it is very critical and sensitive during training. With the hyperparameters configured, the researchers run the training with 5 iterations of Gmall and Ochoa intersection of with different levels of traffic.

4.3 Training Results

Figure 4 shows the performance of DQN applied in Gmall and Ochoa intersection with a moderate level of traffic. The rewards are easily increasing all throughout the training. Figure 5 shows the performance of CMA-ES applied in Gmall and Ochoa intersection with a moderate level of traffic. The rewards are easily increasing all throughout the training, but Gmall got higher rewards than the Ochoa run.

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

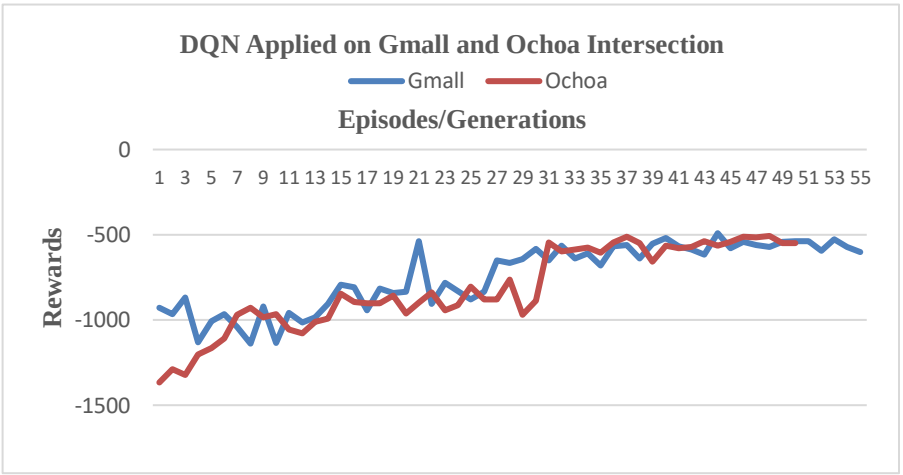

Figure 4. DQN Training Run on Ochoa and Gmall Intersection

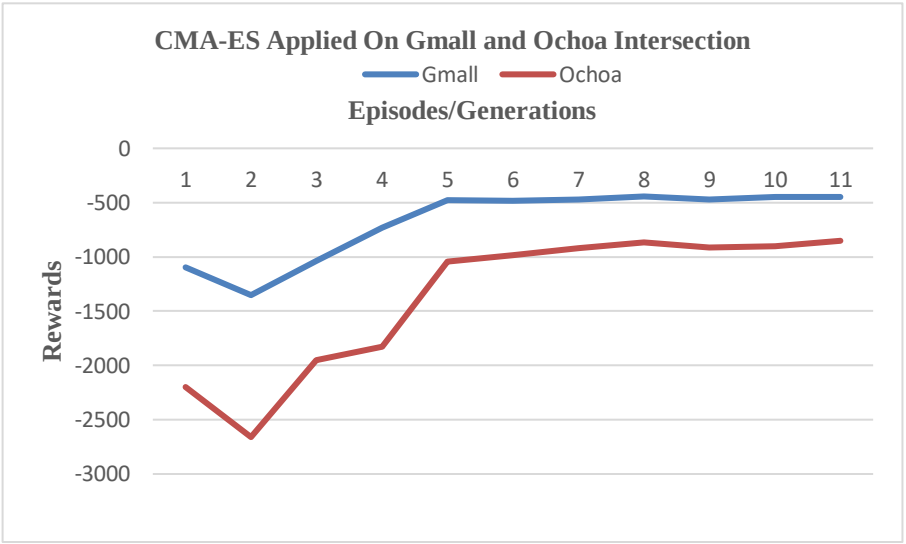

Figure 5. CMA-ES Training Run on Ochoa and Gmall Intersection

The environment gives rewards for every episode/generation based on the number of vehicles in queue and weighted acceleration, i.e., vehicles in queue that have a slow acceleration like tricycles contribute more to slow traffic (large negative reward) while vehicles like sedans can accelerate faster during go time and hence contributes less to traffic (small negative reward). In this simulation, a very large negative reward can mean that the traffic is almost at a standstill. It can be observed from the results that both DQN and CMA-ES achieve an optimal reward of -500. It can also be observed from the plot that the reward signals become less negative as the

ISSN- 2394-5125 VOL 07, ISSUE 15, 2020

training/learning progresses; hence both networks indeed learn an adaptive control policy to keep the traffic flowing and avoiding gridlocks. Table 2 shows the percentage decrease of the waiting time of the vehicles in both intersectionsbased upon the fixed-time results (bigger is better). The majority of all runs in both intersections applying DQN and CMA-ES decreases the average waiting time of the vehicles.

Table2.Decrease of vehicle queue time as compared to fixed-time controller

	Gmall	**Ochoa**
DQN	16.50%	27.80%
CMA-ES	25.90%	33.50%

5.0 CONCLUSION AND FUTURE WORK

The researchers conclude that the traffic simulation implemented via Unity works well and is a great platform for studying different policy search methods for developing an optimal traffic control system. The traffic lights work well as intended, and it mimics the behavior of the traffic light in Butuan City, which is cyclic all in one go (forward and left turn are simultaneous) behavior. The researchers also conclude that the deep q-network and direct policy network are able to control the traffic light agent in Unity 3D. The agent syncs well with the action provided by the network, and it also gives back observations to the network. In the comparison of the performance of algorithms, CMA-ES performs better than DQN in terms of a decrease in the waiting time of vehicles as compared to a fixed-time traffic controller. With this, it can also be concluded that both value iteration algorithms such has DQN and direct policy search algorithms such as CMA-ES achieve an optimal policy that is better than a fixed-time controller. The future work of this study is to apply several other Deep RL methods or algorithms in the simulation. Additionally, adding several vehicle types must also be considered. Finally, non-cyclic action of traffic light can also be implemented and tested.

REFERENCES

[1] Liang, X., Du, X., Wang, G., & Han, Z. (2018). Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks.XX(Xx), 1–11. https://doi.org/10.1109/TVT.2018.2890726

[2] Drugan, M. M. (2019). Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. Swarm and Evolutionary Computation.https://doi.org/10.1016/j.swevo.2018.03.011

[3] Wang, Y., Yang, X., Liang, H., & Liu, Y. (2018). A Review of theSelf-Adaptive Traffic Signal Control System Based on Future Traffic Environment.Journal of Advanced Transportation.

[4] Szepesvari, C., & Littman, M. L. (1996). Generalized Markov decision processes: dynamic-programming and reinforcement-learning algorithms. Technical Report.https://doi.org/10.1.1.39.5887

[5] Jaakkola, T., Singh, S. P., & Jordan, M. I. (1995). Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems.In MIT Press.

[6] Touhbi, S., Babram, M. A., Nguyen-Huu, T., Marilleau, N., Hbid, M. L., Cambier, C., & Stinckwich, S. (2017). Adaptive Traffic Signal Control: Exploring Reward Definition for Reinforcement Learning. Procedia Computer Science.www.doi.org/10.1016/j.procs.2017.05.327

[7] El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2014). Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. Journal of Intelligent Transportation Systems: Technology, Planning, and Operations, 18(3), 227–245.

[8] Van Hasselt, H., &Wiering, M. A. (2007).Reinforcement learning in continuous action spaces.Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL 2007.https://doi.org/10.1109/ADPRL.2007.368199

[9] Arel, I., Liu, C., Urbanik, T., & Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control.IET Intelligent Transport Systems.

[10] Wang, Z., Schaul, T., Hessel, M., & Lanctot, M. (2016). Dueling Network Architectures for Deep Reinforcement Learning Hado van Hasselt.International Conference on Machine Learning.

[11] Mannion, P., Duggan, J., & Howley, E. (2016). An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control. In Autonomic Road Transport Support Systems.https://doi.org/10.1007/978-3-319-25808-9_4

[12] Igel, C. (2003). Neuroevolution for reinforcement learning using evolution strategies.2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings. https://doi.org/10.1109/CEC.2003.1299414