Distributed Fleet Control with Maximum Entropy Deep Reinforcement Learning

Takuma Oda and Yulia Tachibana
DeNA Co., Ltd.

Abstract

In the context of modern vehicle fleets, such as ride-hailing platforms and taxi companies, the ability to proactively dispatch vehicles is instrumental in reducing passenger waiting time and unoccupied cruising time, improving driver profit and decreasing environmental and traffic impact. Among the complex dynamics of fluctuating demand, supply, and traffic conditions inherent in vehicle dispatch, which are almost impossible to fully model explicitly, model-free approaches have shown marked strengths. We present a framework which extends the model-free DQN formulation with soft-Q learning entropy maximization and graph-based diffusion convolution. Comparison against a non-diffusive 'hard' formulation shows significant improvement across several metrics, such as passenger waiting times.

1 Introduction

As global adoption of ride-hailing services continues to rise, the ability to optimize vehicle resource dispatch confers benefits to drivers in reducing idle cruising time and increasing productivity, and shows promise in controlling negative impacts on the environment in terms of traffic congestion. End-users of these services furthermore benefit from increased availability and decreased waiting time when hailing a ride. In addition to these immediate benefits, the necessity for dispatch optimization is further urged by the increasing automation of vehicles, which suggests that soon, fleet vehicles may be making their own dispatch decisions.

1.1 Related Work

Widespread adoption of in-car GPS modules has allowed for estimation of pickup demand in real time, giving rise to models predicting where taxis should be dispatched [1–5]. However, the dynamics of dispatch optimization continue to present many challenges, based on the requirement for real-time decision-making by thousands of vehicles competing for passengers under constantly fluctuating traffic and demand conditions, and the difficulty in evaluating trade-offs among multiple possible target locations. It is often intractable to accurately model all the dynamics involved in such problems due to the many probabilistic aspects involved - towards which model-free reinforcement learning (RL) approaches show promise [6].

Although RL approaches have shown moderate success in optimizing vehicle dispatch in dynamic environments, many formulations are limited in their power to represent realistic road situations, abstracting away key aspects of the real-world spatial and temporal dynamics. In particular, many models abstract out the graph-based architecture of the road network to overly simplified grid models [15], resulting in unrealistically coarse spatial resolution and overlooking the sequential dependencies inherent in traveling down a connected real-world road network.

In addition, most existing RL formulations for the dispatch problem have been based on the assumption that the environment is fully observable by all agents and processed in real-time. This is a crucial difference from reality, where full knowledge about the states and locations of all other vehicles is almost never available to a vehicle, and syncing of information cannot happen instantly.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Entropy-maximizing models such as soft Q-learning (SQL)[10] offer advantages over deterministic ones in allowing for stabilizing perturbations under uncertain dynamics.

1.2 Our Contributions

We present D2C (Distributed Diffusive Control), a framework in which we expand the model-free distributed DQN formulation to maximize entropy in the agents' learning policy with SQL and diffusion convolution over a graph. We make a case that our framework leads to superior performance on multiple metrics over deterministic DQN frameworks [13] and those not including graph-based diffusion, making the following contributions:

- To the best of our knowledge, this work is the first to demonstrate that soft Q-learning is effective for distributing vehicle resources in a not fully observable environment under uncertain demand and dynamics, reducing passenger waiting time by 16.4% using a realistic simulation architecture.
- We design an efficient, novel neural network architecture leveraging the graph structure of the road network to improve overall system performance.

In the following sections we formally define the taxi dispatch problem, then introduce our DQN and diffusion convolution architecture in section III. Our evaluation methods and results are explained in sections IV and V, respectively.

1.3 Problem Definition

Our formulation is based on a control center which sends dispatch directions to a large number of geographically distributed vehicles in real time, and to which passengers make pickup requests using an app. Passengers may not be matched when there are no taxis present in their vicinity, which is something we try to minimize by having taxis learn in which areas they should aim to be.

Following a standard RL formulation, we represent fleet vehicles as agents, each of which observes the state of the environment s and chooses an action a according to a policy $\pi(a|s)$ at each time step. We define the global space in which the agents operate as a grid world of size M_x by M_y with closed borders, which corresponds to the geographic service area. From the agents' perspective, all knowledge of location and movement is abstracted to grid-cell resolution, and all actions are taken between two grid cells agnostic of the underlying geographic coordinates; however, in the interest of preserving the dynamics of a real-world road network graph, inside our simulator we implement this as movement on a geographical map – with routes calculated from the agent's current location to the representative point of the target cell in which the destination coordinates lie.

A vehicle's state is defined by $s = (x, y, t, X_{t-\delta t}, W_{t-\delta t})$. Here, (x, y) are the current grid coordinates, t is the current time, $W_{t-\delta t}$ and $X_{t-\delta t}$ are representations of the latest global supply and demand for the upcoming time interval. In order to approximate reality, we assume that vehicles are not aware of the real-time locations of all other vehicles, and as it takes time δt to process data and calculate states, processing is not fully real-time but rather works in mini-batch.

Each vehicle receives an order to perform an action a, corresponding to a dispatch order from the control center in the real world. The action space of an agent is a size A_x by A_y neighborhood of grid cells, and is updated for each agent at each step based on its current coordinates. Each action takes τ minutes, defined as the road-network travel time to a grid cell within the action space. A new dispatch order is made only after the vehicle has completed the previous action, representing the fact that giving orders too often in the real world could lead to erratic or unsafe driving resulting from too many trajectory changes.

The agents' goal is to maximize expected discounted return $R_t = \sum_{t=0}^{t_{\max}} \gamma^t r_t$, where $\gamma \in [0, 1)$ is the discount factor. We express the reward function for an individual agent as:

$$r_t = f_t - w_t - c_t, \tag{1}$$

where f_t is the collected fare during time slot t, and w_t and c_t are the working costs and cruising costs, respectively. While not explicitly incorporated into the reward, decreasing the reject rate of ride requests and reducing passenger waiting time is another objective, achieved through distributed decision-making by agents.

2 System Design

2.1 Reinforcement Learning Architecture

The basic architecture of our formulation is based on a Deep Q-network (DQN), which enables learning the action-value function approximator directly from the environment using a deep neural network. The instabilities or divergence resulting from using nonlinear functional approximators such as neural networks to represent the Q-function are successfully addressed through experience replay, and we account for the issue of nonstationarity due to constantly changing environment dynamics by removing obsolete information from replay buffers, and learning estimates of other agents' current policies as inferred from their behavior. However, as the number of agents and the action space increase, such as in the case of a large fleet with a wide service area, scaling the DQN presents increasing computational costs. We addressed this by having each agent learn the optimal policy separately and share experience memory, borrowing from the concept of independent Q-learning (IQL) - a bottom-up agent-centric distributed approach in which every vehicle agent learns its own Q-function, instead of explicit top-down coordination among all agents [7].

Many existing RL implementations operate under the assumption that the optimal policy under full observability is deterministic [12]; however, real-life traffic scenarios are in principle not fully observable in real-time by all agents, and additionally involve highly complicated demand and traffic dynamics, making it difficult to represent the evolution of road conditions and the resulting driver policies as deterministic processes. In complex environments such as these, stochastic policies have shown to offer more robustness to perturbations and local maxima by improving the quality of exploration, as well as producing decent performance even under multi-modal objectives and composite actions, such as in the case of a vehicle having to choose among multiple target locations. The stochastic policy is expressed as an energy-based model controlling the amount of entropy in the agent's policy. Our formulation is based on the soft Bellman equation:

$$Q^*(s_t, a_t) = r_t + E_{s_{t+1}\sim p}\left[V^*(s_{t+1})\right] \tag{2}$$

$$V^*(s_t) = \alpha \log \int_A \exp\left(\frac{1}{\alpha} Q^*(s_t, a')\right) da'. \tag{3}$$

The optimal policy is given by:

$$\pi(a_t|s_t) = \exp\left(\frac{1}{\alpha}(Q^*(s_t, a_t) - V^*(s_t))\right). \tag{4}$$

While local entropy-maximizing methods such as Boltzmann exploration [18] and policy gradients are effective in increasing entropy at individual action steps, soft Q-learning as in our formulation extends this concept to maximizing entropy over entire state trajectories, aiming to reach future high-entropy states, by expressing the optimal policy itself as inherently stochastic [10].

2.2 Neural Network Architecture

Graph convolutions provide a powerful generalization of the convolutional neural network (CNN) framework to non-Euclidean, directional graph-based architectures such as road networks. We utilize a version of graph convolution in our work called the diffusional-CNN, which builds a latent representation of underlying features similar to that in grid-based CNN formulations, but scanning diffusively along neighboring nodes in a graph through random bidirectional walks. The diffusion convolution (DC) operation, which learns latent representations of graph-structured data[11], is defined in Equation 5:

$$X \star g f_\theta = \sum_k \left(\theta_{k,1}(D_{\text{out}})^k + \theta_{k,2}(D_{\text{in}})^k\right) X \tag{5}$$

Above, the matrices D_{out} and D_{in} are the transition matrices for diffusion in outward and inward directions, respectively, and θ represents the diffusion filter parameters.

Input to the network at each time step consists of pre-processed heat maps on a grid the same size as the global service area. The global demand map consists of a statistical demand profile component

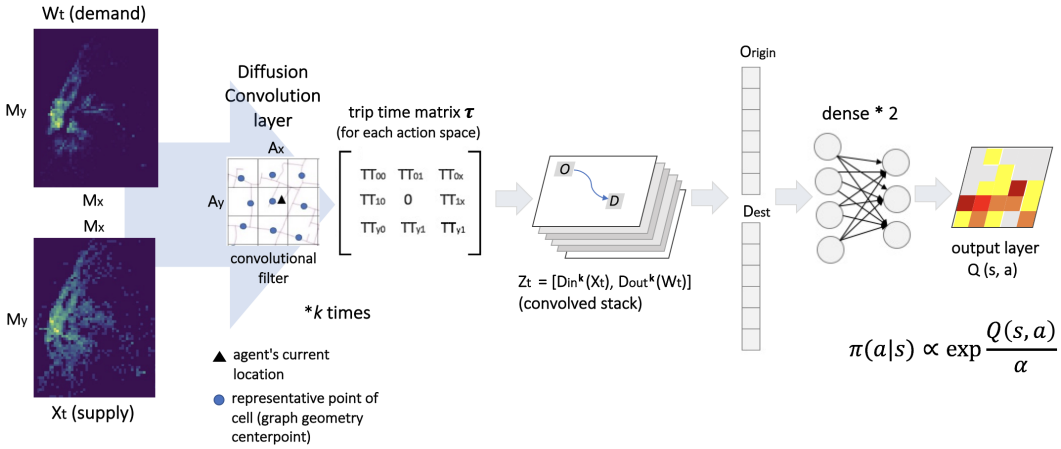

Figure 1: D2C architecture

Algorithm 1: D2C learning algorithm

for $t = 0 : T_{max}$ **do**
  Get idle vehicles $\mathcal{I}_t$ from Simulator;
  if $t \mod T_d = 0$ **then**
    Update current global demand W_t and supply X_t ;
    Compute diffusion convolved map Z_t;
  end
  for $i \in \mathcal{I}_t$ **do**
    Extract i-th vehicle's action space $\mathcal{A}_t^{(i)}$ reachable from location $l_t^{(i)}$ within $\tau_{\max}$;
    Set $s_t^{(i)} = (t, l_t^{(i)}, Z_t)$;
    Get i-th vehicle's previous state (t_p, R_p, s_p, a_p) from cache $\mathcal{C}$;
    Compute reward $r_t^{(i)} = R_t^{(i)} - R_p$;
    Store transition $(t_p, s_p, a_p, r_t^{(i)}, t, s_t^{(i)})$ in replay memory $\mathcal{D}$;
    Overwrite state action pair $(t, R_t^{(i)}, s_t^{(i)}, a)$ in $\mathcal{C}$;
    Compute $q_{a'} = Q^\theta(s_t^{(i)}, a')$ for all $a' \in \mathcal{A}_t^{(i)}$;
    With probability ϵ select a random action $a \in A$ otherwise sample $a \sim \pi^\theta(a'|s) \propto \exp(q_{a'}/\alpha)$;
    Select a destination coordinate d of the action a;
    Add (i, d) to the dispatch solution;
  end
  Update vehicles' state according to dispatch solutions on Simulator;
  Sample random minibatch of transitions $(t_j, s_j, a_j, r_j, t'_j, s'_j)$ from $\mathcal{D}$;
  Compute $y_j = r_j + \gamma^{t'_j - t_j} \text{softmax}_{a'_j} \hat{Q}^{\theta^-}(s'_j, a'_j)$ for all j;
  Perform a gradient descent step on $\sum_j (y_j - Q^\theta(s_j, a_j))^2$ with respect to θ;
  Every E steps reset $\theta^- = \theta$;
end

and a dynamic component representing demand information up to the current moment, which are both used as features in the demand diffusion model.

All features pass through a DC layer where graph diffusion convolution is carried out k times with a filter of dimensions $M_x \times M_y \times A_x \times A_y$. The diffusion probability can be defined by the Radial Basis Function of road network trip time τ between two grid cells, calculated among the coordinates of the cells' representative points – ie. the centermost point on the graph geometry inside the cell:

$$p \propto \exp(-\tau^2/\sigma^2). \tag{6}$$

These convolved layers are composed into a 3-D tensor Z_t as below:

$$Z_t = [(D_{\text{in}})X, (D_{\text{out}})W \ldots (D_{\text{in}})^k X, (D_{\text{out}})^k W]. \tag{7}$$

D_{in} and D_{out} represent the diffusion processes directed inward and outward from each grid cell, corresponding to vehicle influx and outflux from each location. From this, we choose a set of

destination coordinates (x', y'), to which a route is calculated from our origin coordinates (x, y), which is repeated for all grid cells in the current action space. The feature vectors at (x, y) and (x', y') then pass through fully-connected layers and output action values $Q(s, a)$ for an action a. Note that the diffusion operations are executed only when global supply and demand are updated, and once Z_t is updated, this data can be shared by all vehicles, indicating that no diffusion operation is needed at inference time.

Although in the local action space for each vehicle our diffusion is based on the graph architecture of the road network, globally we represent diffusion as a process among grid cells, rather than on the fully-connected graph. Diffusion on the entire fully-connected road network graph at each time step is not only realistically unnecessary, as effects from nearby nodes drop off greatly with distance, but also computationally intractable. We find that approximation of the current action space for several time steps as grid cells, while conserving the road connections within the span of these cells as a graph representation, to be a sufficiently effective method for representing diffusion dynamics.

The overall D2C learning algorithm is outlined in Algorithm 1. Although each agent learns separately, we implement a replay buffer which is shared among all agents, and one network is learned and used for inference decisions by all agents.

3 Experiments and Results

Simulator Design: D2C is based on a realistic taxi fleet simulation framework as illustrated in Figure 2, which simulates vehicles' state changes by either matching requesting customers with vehicles, or by proactively dispatching vehicles to locations. In the scope of this work we focus only on optimization of proactive dispatch, and assume that matching between a customer and a vehicle is determined by the greedy nearest neighbor algorithm, implemented in the simulator. At each time step, D2C computes the next actions for vehicles which have dropped off their passengers or arrived at their dispatched locations within this time slot. Each trip between two locations is designated via shortest path and its travel duration calculated by the Open Street Routing Machine (OSRM)[17]. Although actions are represented as happening among grid cells, the simulator internally uses the graph representation of the road network for routing.

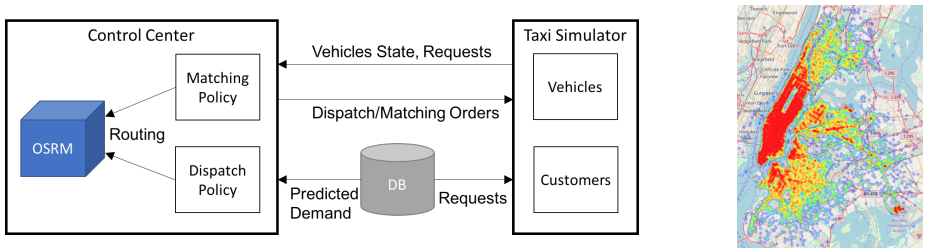

Figure 2: The modular architecture of D2C simulation framework (left) and geographic demand distribution in the experiment area in NYC (right)

Implementation Details: We base our implementation on NYC Taxi and Limousine Commission trip records from June 2016 [16]. While each pickup location in this dataset represents where a passenger previously hailed a taxi on the street, we assume the distribution of demand is similar when the passenger uses a mobile application. The entire service area is divided into a 52×68 grid of regions, each of which is around $500 \times 500\text{m}^2$. Each vehicle can move at most 7 regions horizontally or vertically from its current region in a single action, resulting in a 15×15 map of possible destination regions. We compute trip time for all possible actions based on OSRM's estimated travel time between origins and destinations and construct an action trip time map τ, which is also used for the diffusion filters. Diffusion convolution uses $k = 3$ layers and $\sigma = 450$ seconds, and given the size of the immediate action space of 7 grid cells, this corresponds to a diffusion process looking $(3 + 1) \times 7.5 = 30$ minutes forward for the current state. We trained the Q-network for two weeks of simulation using RMSProp optimizer. The behavior policy during training was ϵ-greedy with ϵ annealed linearly from 1.0 to 0.01 over the first 5,000 training steps. All simulations run with a 1-minute simulation step and 8,000 vehicle agents. We conducted another one week of simulation for the evaluation of each model with a test dataset containing 3 million trip records.

Table 1: Summary of results

Policy	revenue ($/h)	cruising time (h/day)	reject rate (%)	waiting time (s)
Wait	28.70	1.67	17.06	145.33
Random	30.48	5.13	12.09	144.86
Hard DQN-A	32.48 ± 0.05	4.49 ± 0.16	4.30 ± 0.11	96.68 ± 1.53
Hard DQN-D	32.69 ± 0.07	4.53 ± 0.17	4.02 ± 0.05	94.12 ± 1.18
Soft DQN-A	32.53 ± 0.03	4.47 ± 0.13	4.13 ± 0.09	82.88 ± 1.42
Soft DQN-D	$\mathbf{32.73} \pm 0.06$	$\mathbf{4.42} \pm 0.23$	$\mathbf{3.97} \pm 0.08$	$\mathbf{82.80} \pm 1.88$

Performance Comparison: In order to evaluate the benefits of diffusion convolution and maximum entropy RL, the performance under each of the following policies is extensively evaluated in the simulation. Each situation below depicts the agent's policy when it has become idle (s.a. after passenger drop-off):

- **Wait:** vehicles stay at their locations and wait to be assigned a passenger
- **Random:** vehicles move randomly to a reachable grid cell
- **Hard/Soft DQN-A:** vehicles move to a location determined by a DQN with 3 average pooling layers with stride-1 trained with $\alpha = 0$ (hard) or $\alpha = 0.2$ (soft)
- **Hard/Soft DQN-D:** vehicles move to a location determined by a DQN with 3 travel time diffusion convolution layers trained with $\alpha = 0$ (hard) or $\alpha = 0.2$ (soft)

We defined four metrics for performance evaluation: request reject rate, average revenue per hour, average cruising hours per day and average passenger waiting time per request. Table 1 summarizes the simulation results across each metric. Compared to the Random policy, Soft DQN-D achieves significant improvement across all metrics, decreasing passenger waiting time by 14.4% while also increasing revenue and decreasing reject rates. Considering the increased performance in terms of revenue and reject rate of DQN-D (diffusion) compared to that of DQN-A (average), it can be said that road network diffusion allows for better and more granular demand prediction, and selection of more profitable actions.

It can also be seen that both of the soft DQN models significantly contributed to reducing passenger waiting times. This can be explained by the fact that the stochastic policy avoided local maxima in distribution and led to a wider and smoother geographic distribution of vehicle agents as compared to that under deterministic policies, resulting in vehicles being more likely to end up within reachable distances for requesting passengers and being able to travel to pick-up locations faster.

4 Conclusion

The distributed soft Q-learning framework with graph-based diffusion convolution presented in this work has shown significant advantages over standard "hard" and non-diffusive DQN, decreasing passenger waiting times and reject rates while increasing driver revenue. We believe this makes a case for the benefits that entropy maximization offers in handling the uncertainties inherent a partially-observable multi-agent environment, and the strengths of graph diffusion in accurately creating latent representations of features in non-Euclidean spatial relationships. It remains important to explore the theoretical impacts of diffusion convolution and entropy maximization, and further work will investigate the concrete reasons for the contributions of these elements to each of the mentioned performance metrics, so as to optimize the trade-offs between waiting time and occupancy rate.

References

[1] Miao, F., Han, S., Hendawi, A. M., Khalefa, M. E., Stankovic, J. A. & Pappas, G. J. Data-driven distributionally robust vehicle balancing using dynamic region partitions. In *ICCPS '17, Proceedings of the 8th International Conference on Cyber-Physical Systems*, pages 261-271, 2017.

[2] Zhang, D., He, T., Lin, S., Munir, S. & Stankovic, J. A. Dmodel: Online Taxicab Demand Model from Big Sensor Data in a Roving Sensor Network. In *IEEE International Congress on Big Data*, pages 152-159, 2014.

[3] Li, B., Zhang, D., Sun, L., Chen, C., Li, S., Qi,G. & Yang Q. Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *the IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 63-68, 2011.

[4] Jauhri, A., Joe-Wong, C. & Shen, J. P. On the real-time vehicle placement problem. In *the 31st Conference on Neural Information Processing Systems (NIPS) Workshop on Machine Learning for Intelligent Transportation Systems*, 2017.

[5] Yuan, N. J., Zheng, Y., Zhang, L. & Xie, X. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. In *IEEE Transactions on Knowledge and Data Engineering archive*, 25(10):2390–2403, 2013.

[6] Oda, T. & Joe-Wong, C. MOVI: A Model-Free Approach to Dynamic Fleet Management. In *IEEE INFOCOM*, 2018.

[7] Tan, M. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. *Readings in Agents*, pages 487–494, 1997.

[8] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H. S., Kohli, P. & Whiteson, S. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (PLMR) 70*, 2017.

[9] Atwood, J. & Towsley, D. Diffusion-Convolutional Neural Networks. In *the 30th Conference on Neural Information Processing Systems (NIPS)*, 2016.

[10] Haarnoja, T., Tang, H., Abbeel, P. & Levine, S. Reinforcement Learning with Deep Energy-Based Policies. In *Proceedings of the 34th International Conference on Machine Learning (PLMR) 70*, 2017.

[11] Li, Y., Yu, R., Shahabi, C. & Liu, Y. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*, 2018.

[12] Sutton, R. S. & Barto, A. G. In *Reinforcement Learning: An Introduction.* pages 185–186, 2005.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. Human-level control through deep reinforcement learning. In *Nature* 518(7540):529-33, 2015.

[14] Ziebart, B. D., Maas, A. L., Dey, A. K. & Bagnell, J. A. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proc. of ACM UbiComp*, pages 322–331, 2008.

[15] Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W. & Ye, J. Large-Scale Order Dispatch in On-Demand Ride-Sharing Platforms: A Learning and Planning Approach. In *KDD '18 Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 905-913, 2018.

[16] New York City Taxi and Limousine Commission. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`

[17] Open Street Routing Machine `http://project-osrm.org/`

[18] Sallans, B. & Hinton, G. E. Reinforcement learning with factored states and actions. In *Journal of Machine Learning Research* Volume 5, pages 1063–1088, 2004.

[19] van Hasselt, H., Guez, A. & Silver, D. Deep Reinforcement Learning with Double Q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016.