Drivers As Agents: A Reinforcement Learning Approach to Model Drivers

Ali AlNasser [1] **Abdulrahman AlFawzan** [1] **Mohammed AlBarrak** [2] **Fahad AlSaawi** [2] **Nouf AlZamel** [2] **Muhannad AlZahrani** [2]

Abstract

In this paper, we present a practical reinforcement learning approach simulating a driver's behaviour in accepting/rejecting incoming delivery orders. Further, we discuss shortcomings of distance based pricing strategies. We utilized Jahez order dataset to train the agent and compare the results with actual driver behaviour. We limit our work to the Riyadh city orders. Our work makes the following contribution: (1) Modelling a realistic drivers' reward function, the outcome of which is an agent policy that simulate realistic behaviour. (2) We develop an environment to study the interplay dynamics of new pricing strategies with acceptance policies. (3) We introduce a new pricing strategy to decrease drivers rejection rate.

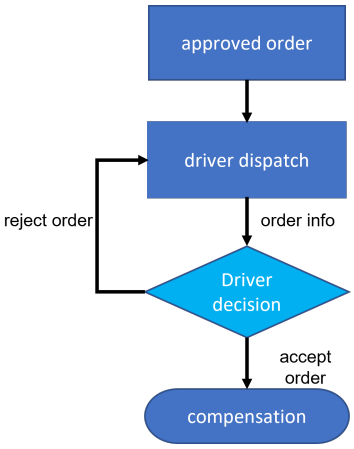

Figure 1. Approved order process: after acceptance from the branch, the order goes to the driver dispatching system in order to find a driver to accept the order. The dispatching system cycles through drivers until a driver accepts the order. Once a driver is dispatched, they are given a delivery compensation

1. Introduction

The journey of Jahez order starts with an order submission from the client (customer) to a Jahez business partner through Jahez application. Once the order is approved by the business, it goes through driver dispatching process, where a driver is presented with an order to deliver. The driver is shown information about the order consisting of customer and business locations and the delivery compensation. Based on which the driver takes the decision of accepting or rejecting the order, if the driver accepts the order, they are given a delivery compensation which is computed via a pricing strategy. If the order is rejected, it is dispatched to another driver. The iteration ends when a driver accepts the order. Figure 1 shows a flow chart of this journey. The order may remain in the loop for any period of time. Resulting in an unacceptably long delay in order delivery. The goal of this work is to minimize the time spent to assign drivers to orders through introducing an optimal pricing strategy. The following sections addresses this problem and proposes a solution using a reinforcement learning approach.

2. Background

The field of reinforcement learning has been in development for many years. The interest has peaked after the success of Deepmind's AlphaGo system(Silver et al., 2016). Since then, problems that are typically tackled by the field are dominated by games (Brockman et al., 2016; Lanctot et al., 2019; Nichol et al., 2018) and control tasks (Yu et al., 2019; Tunyasuvunakool et al., 2020). Control tasks problems made its way into the industrial applications via the robotic arm (Jafari-Tabrizi & Gruber, 2021). A team of researchers has developed a library (AI4finance) to train models for trading stocks(Liu et al., 2021). In commercial applications, dynamic pricing is an area of interest(den Boer, 2015), a team from Alibaba created a reinforcement learning model to dynamically price items on their e-commerce platform. (Liu et al., 2019).

Commercial applications powered by reinforcement learning are very limited. To our knowledge, no work has been done locally in this area. We hope that this paper starts the inquiry of practical reinforcement learning application on local data in unexplored sectors.

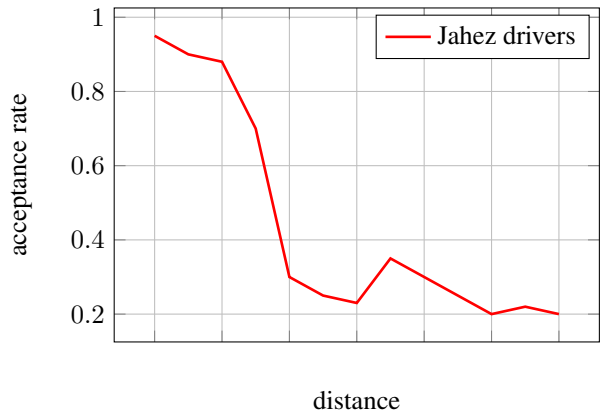

Figure 2. The average acceptance rate of high performers falls as the distance increases, the exact distance is omitted for privacy purposes

3. Exploratory Data Analysis

Incoming orders to Jahez have a high variance of *acceptance rate* which result in a volatile customer experience. preliminary analysis showed that most orders are delivered by *high performers*, therefore, this work considers this subset as the population of drivers. Further, the variance in acceptance rate is attributed to multiple factors: distance, time, branch and customer location. With distance being the primary factor. Figure 2 shows the relationship between acceptance rate and distance.

The dataset is comprised of orders containing the following information:

- business location
- customer location
- business to customer distance
- datetime stamp
- delivery time
- delivery compensation
- rejection count

In our experiments, we chose the following features as the deciding factors of the driver decision:

- **branch and customer location:** some location pairs are more desirable than others
- **timestamp:** gives insight to rush hours and number of orders per hour
- **distance:** as distance increases, drivers are less likely to accept orders

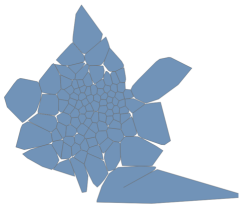

Figure 3. clustering order longitudes and latitudes into polygons representing zones. The zones are assigned IDs to reference them as locations

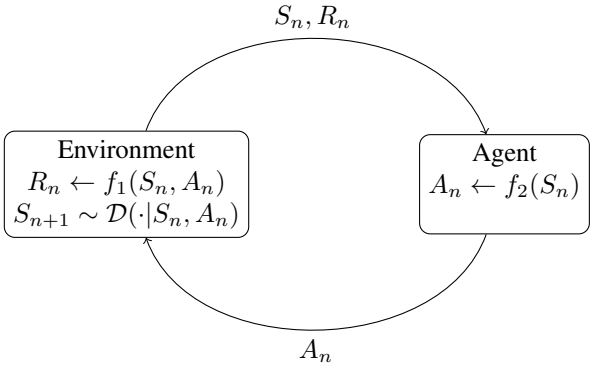

Figure 4. interaction between environment and agent: the environment send a state (S) to the agent, the agent respond by computing and sending an optimal action (A) to the environment which is used to compute the immediate reward (R) and generate the next state.

3.1. Clustering

It is intractable to use the locations as pairs of latitude and longitude. We therefore segment the city into multiple zones via a clustering algorithm and consider the zones as locations of branches and customer. We used customer locations to segment Riyadh city into multiple zones by clustering order locations. The number of zones is arrived at to fit business needs as well as multiple empirical iterations of different zone sizes. Figure 3 shows a map of Riyadh zones. The IDs of the zones are then used as proxy for the order locations.

4. Reinforcement Learning

This section covers the reinforcement learning setting for our application. The environment will represent orders as states, whereas the agent will represent the driver. The environment will produce a state (order) for the agent (driver). The agent will reply by taking an action which the environment uses to compute the reward and generate the next state. Figure 4 shows the relationship between the environment and agent. The functions f_1, f_2 will be discussed in subsequent sections

4.1. Environment

The environment represents the state to the agent as a vector of the following form:

$$\text{state } S = \begin{bmatrix} \text{branch zone} \\ \text{customer zone} \\ \text{day} \\ \text{hour} \\ \text{distance} \end{bmatrix} = \begin{bmatrix} bz \\ cz \\ D \\ H \\ d \end{bmatrix} \tag{1}$$

In addition to the listed feature. The environment keeps track of extra features that will be used to evaluate the reward function:

- delivery time DT
- delivery compensation dc
- timestamp T

.The environment samples a state uniformly conditioned on the previous state as follows:

$$\begin{aligned} S_1 &\sim \mathcal{U}(\cdot) \\ S_{(n+1)} &\sim \mathcal{U}(\cdot | bz_{(n+1)} = A_n bz_n + (1 - A_n) cz_n, \\ D_{(n+1)} &= D_n, \\ T_{(n+1)} &> T_n + dt_n A_n, \\ T_{(n+1)} &\leq T_n + dt_t A_n + \delta t_n) \end{aligned} \tag{2}$$

Where $A_n = 0$ if the agent rejects the order and $A_n = 1$ if accepted, dt_n denotes delivery time, and δt denotes the time window to sample from. Note that dt_n and δt are known in the environment, but are not known to the agent. Each episode represent one day, The environment keeps generating states until the end of the episode is reached.

4.2. Agent

The agent will learn a policy to take the optimal action for every state it receives. The optimal policy (represented in figure 4 as f_2) is achieved by maximizing the reward function over the episode. It will be represented as a neural network, taking a state as an input and outputting an action.

4.3. Reward function

The reward function is the primary objective for the agent to maximize. We modelled the reward function to have two major components: the delivery compensation and the delivery time. It was also important to model the cost for the agent to deliver an order. That is, the time taken to deliver the order. we propose a measure of *minute value* (mv), which is the cost of one minute of the agent's time.

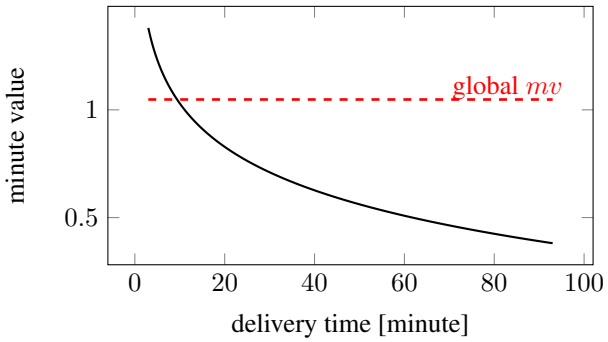

Figure 5. minute value vs. time. the global mv value is calculated as $\frac{\sum dc}{\sum dt}$

We further note that the minute value is sub-linear: the cost of delivering two 10 minutes orders is higher than delivering one 20 minutes order. The general shape of the function can be seen in figure 5. The model to compute *mv* is removed for privacy purposes.

The driver's time cost (dtc) is calculated as

$$dtc = dt \cdot mv(dt) \tag{3}$$

Finally the reward function is the difference between the received compensation and the time cost:

$$\text{reward } R_n = dc_n - dtc_n \tag{4}$$

4.4. Learning Iteration

In every step n, the environment sends a state to the agent, the agent, then computes a forward pass of its Q network. The action is taken to be the action that maximize the Q-value function:

$$a_n = \pi(S) = \arg\max_a Q(S_n, a) \tag{5}$$

Where $Q(S, A)$ represent the agent's estimation of the Q-value function at state S when taking action A, π denotes the agent's action policy. We use temporal differences(Sutton, 1988) to calculate the future reward estimation using Q. Q is updated iteratively to reach the optimal Q^* and policy π^* as shown in equation (6)

$$Q_{n+1}(S, A) \leftarrow (1-\alpha) Q_n(S, A) + \alpha [r + \gamma \max_a Q_n(S', a)] \tag{6}$$

Where α is a learning rate, γ is the future reward discount factor, S' is the state reached after taking action A from state S. This can be interpreted as the value of action A at state S is the previous estimation plus the future values from that state onward to the end of the episode.

Figure 6. area and zones to sample orders from, the label denotes the zone ID

The Q function is represented as a neural network and can be denoted explicitly with its parameters as $Q(S, A|w)$. The loss function is defined as the difference between two iterations of the Q-value function. With the goal to reach an optimal Q-value at the end of training.

$$\begin{aligned} L(w) &= \mathbb{E}_{(s,a)}||Q_{i+1}(s,a|w) - Q_i(s,a|w)||_2^2 \\ &= \alpha\mathbb{E}_{(s,a)}||[r + \gamma \max_a Q_i(s',a|w) - Q_i(s,a|w)||_2^2 \end{aligned} \tag{7}$$

Initialized randomly, we iteratively update the network parameters w via gradient descent:

$$w_{i+1} \leftarrow w_i - \lambda \nabla_w L \tag{8}$$

for some learning rate λ

5. Experiment Setup

In order to model real drivers, some constraints have to be imposed to the experiment environment:

1. our work attempts to address the current behaviour of drivers. We consider the last six months of orders for our dataset to sample orders from.
2. drivers don't wonder around the city, rather, they specialize in specific regions (spanning multiple zones). In our experiment, We consider customer and business locations in the northern Riyadh region. Figure 6 shows the region and zones considered in the experiment.
3. drivers don't spend 24 hours a day delivering orders. Hence, we select 12 hours, ranging from **11am to 11pm**. This range covers both lunch and dinner rush hours, as well as some regular hours. This is a lenient constraint, and can be lifted with some extra consideration of the agent model.

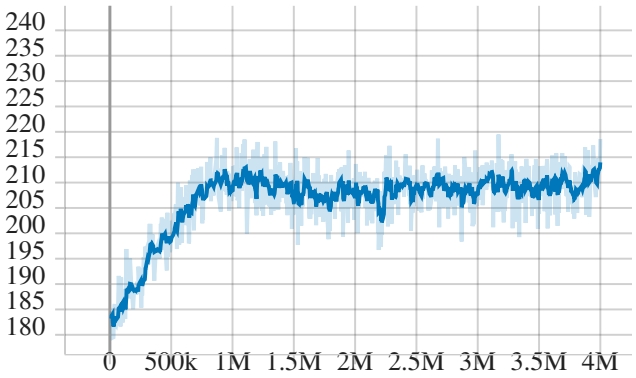

Figure 7. Cumulative reward vs. episode, notice the increase in reward at first, indicating some knowledge has been gained by the agent

The training was carried for 4M episodes, the cumulative reward is logged per episode. With this setup and reward function, the agent was able to learn multiple real driver behaviour patterns, the following section discusses our findings.

6. Results

6.1. Initial results

Figure 7 shows the cumulative reward per episode. We notice an increase of 17% in the reward at the beginning of the training period. This is where the agent has learned to discern differences between orders and selectively pick orders that will maximize the overall reward. The following subsection discusses some of the observed behaviours the agent has learned.

6.2. Agent vs. driver comparison

6.2.1. DISTANCE PATTERN

Since the existing pricing strategy is distance based, we notice a highly correlated pattern of acceptance between the agent and Jahez drivers. Figure 8 shows the acceptance rate vs. distance for both Jahez driver and the simulation agent. The figure demonstrates that the agent learned the compensation scheme and how to maximize the financial reward. We also notice the acceptance of the agent in general is higher than Jahez drivers. This maybe due to the fact that the agent evaluates based on the order info without any downtime whereas the driver may reject an order for other activities irrelevant of the order such as eating, resting etc.

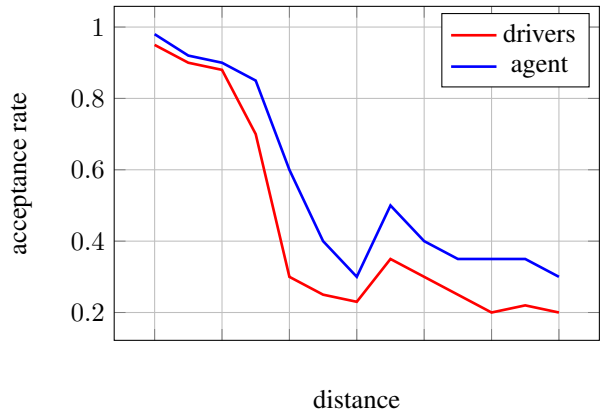

Figure 8. acceptance rate vs. distance, comparison between agent and Jahez drivers

Jahez driver acceptance rate

day	regular hours	rush hours
weekday	1.41	1.01
weekend	0.91	0.66

Simulation agent acceptance rate

day	regular hours	rush hours
weekday	1.19	0.96
weekend	1.05	0.78

Table 1. driver acceptance rate per weekday and hour categories, the values are normalized for privacy purposes

6.2.2. Timestamp Pattern

The agent demonstrates that it learned to be more selective in high demand times compared to low demand times just as observed in Jahez drivers behaviour. The gap between rush hours of weekend and regular hours of weekdays are similar between the agent and drivers. We notice the drivers acceptance of rush hours in weekdays is higher than regular hours of weekends whereas the agent selection is the opposite. This maybe due to the fact that neither the reward function nor the environment setup takes into consideration a cab on number of delivered orders daily, which can be translated into the driver capacity to work or tiresome which occurs on weekends hence its less acceptance for actual drivers. Table 1 summarizes the acceptance rate per day and hour group.

6.2.3. Obtained Knowledge from the Environment

Figure 9 represents the acceptance rate of the agent if the customer resides in that zone. We notice the agent learned that in general delivering orders to rural zones such as 11 and 12 (see figure 6) is not desirable because it limits the number of potential future orders whereas the highly populated zones are close to each other in acceptance because

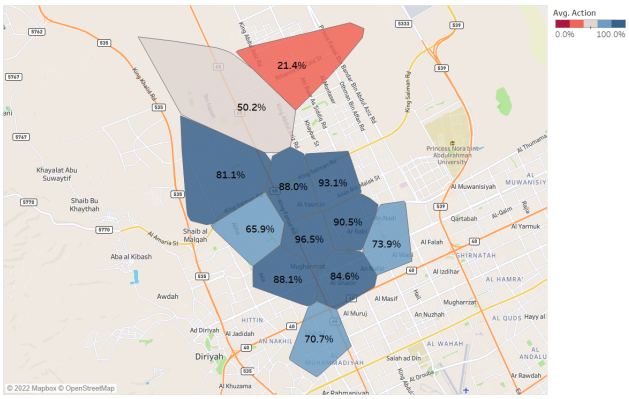

Figure 9. acceptance rate from any branch zone to customer zone. The probability of going to a customer zone averaged over all branch zones. i.e. $P(cz)$

demand and supply are more frequent. We can represent the acceptance rate as the probability of acceptance. Figure 9 shows the following probability

$$P(cz) = \sum_{bz} P(bz \rightarrow cz|bz)P(bz) \tag{9}$$

Figure 10 represents the acceptance rate of the agent if the business location is in zone 1 and the customer resides in each zone. We notice the agent learned that the business is located in zone 1, then zones 4 and 7 are not desirable because the distance is not far enough to compensate for the time wasted in traffic whereas the zones 11 and 12 are well compensated because of their distance and quick access due to highway roads availability. i.e. the difference in time between delivering to zone 4 and zone 12 is not significant, however the compensation difference makes going the extra distance and time more rewarding.

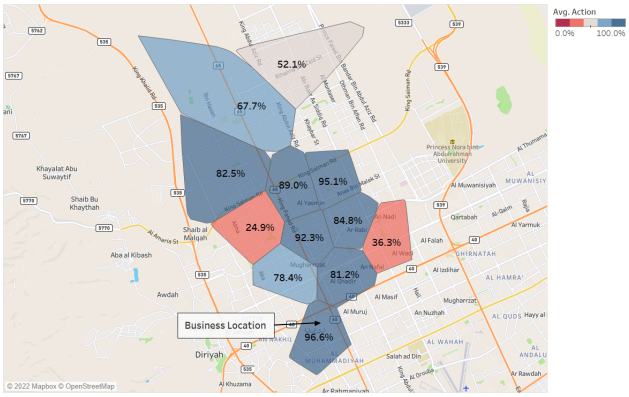

Figure 10. acceptance rate starting from branch zone 1 and going to customer zone. i.e. $P(cz|bz = 1)$

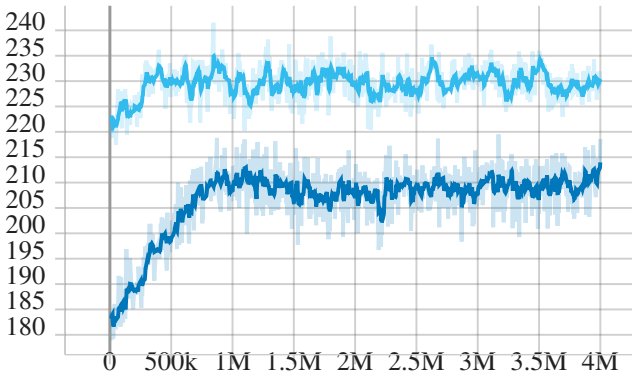

Figure 11. cumulative reward per episode. Comparison between the new pricing strategy (light blue) and original pricing strategy (dark blue)

6.3. New pricing strategy

For a pricing strategy to succeed the driver shouldn't be able to maximize their reward by rejecting some orders in favor of others. Since the agent was able to behave like real drivers in different scenarios (see figure 8) We propose a pricing strategy that is time based rather than distance based. The new pricing strategy can take one of many forms. For instance, a fixed minute value can be set to compute a fixed rate. The delivery cost will be calculated as follows:

$$dc = dte \cdot MV \tag{10}$$

for fixed MV, where dte denotes an estimation of the delivery time. The strategy has been tested on the environment to investigate whether the agent can learn to maximize their reward by rejecting certain orders. Figure 11 shows the reward per episode of the new and original strategy. Note how there is an increase of 4.5% in the reward compared to the increase of 17% with the original pricing strategy, note the number of episodes need to converge to a policy is less than half when compared with the previous strategy. Both remarks indicate that the agent was not able to discern the differences between orders with the new strategy, making all orders indistinguishable as presented to the drivers.

7. Conclusion and Future Directions

The process of assigning drivers to orders for delivery may take longer than acceptable by business standards. This issue may result in potential revenue loss. The current delivery compensation offered to drivers made some orders more favorable than others. In this work, we develop a simulation of the driver as a reinforcement learning agent. The agent has learned to exploit the distance based pricing strategy by selectively accepting and rejecting orders. To address this issue, a strategy is needed where the driver is incentivized to accept all incoming orders, we propose a new pricing strategy under which the agent is not able to favor some orders over others. Further, our work introduces an environment where pricing strategies can be analyzed and assessed in a simulated environment to evaluate pricing strategies with minimum room for exploitation and to mitigate risks of multiple strategy testing experiments in the real environment.

We considered two classes of pricing strategies: distance based and time based strategies. However, there are multiple different pricing strategies to be explored. Of particular interest is a reactive pricing strategy: where the pricing is dependent on the probability of rejection, which will incentivize orders with low likelihood of acceptance to be accepted by drivers. Another avenue to explore is to consider a hybrid approach of multiple pricing strategies.

Another future direction would be to remodel the environment to emphasize different driver behaviours. For instance, our work concentrate on modelling high performers and does not model all drivers, also, our work concentrate on modelling a big city environment, where there is a high density and wide variety of orders in terms of distance, time, etc. Changes in the environment setup will allow for modelling segments of drivers in different cities. Further, the reward function can be modelled to achieve different business objectives. Our work concentrate on modelling the reward function from the driver's financial gain perspective such that orders are equally favorable to drivers. Other objectives may prioritize the number of delivered orders to incentivize drivers to cover more working hours, or prioritize the balance of supply and demand to emphasize driver availability around the city in a multi-agent environment.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016. URL https://www.gymlibrary.ml/.

den Boer, A. V. Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys in Operations Research and Management Science*, 20(1):1–18, 2015. ISSN 1876-7354. doi: https://doi.org/10.1016/j.sorms.2015.03.001. URL https://www.sciencedirect.com/science/article/pii/S1876735415000021.

Jafari-Tabrizi, A. and Gruber, D. P. Reinforcement-learning-based control of an industrial robotic arm for following a randomly-generated 2d-trajectory. In *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pp. 1–6, 2021. doi: 10.1109/COINS51742.2021.9524158.

Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V.,

Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., Vylder, B. D., Saeta, B., Bradbury, J., Ding, D., Borgeaud, S., Lai, M., Schrittwieser, J., Anthony, T., Hughes, E., Danihelka, I., and Ryan-Davis, J. OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019. URL http://arxiv.org/abs/1908.09453.

Liu, J., Zhang, Y., Wang, X., Deng, Y., and Wu, X. Dynamic pricing on e-commerce platform with deep reinforcement learning: A field experiment, 2019.

Liu, X.-Y., Yang, H., Gao, J., and Wang, C. D. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. *ACM International Conference on AI in Finance (ICAIF)*, 2021.

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, Aug 1988. ISSN 1573-0565. doi: 10.1007/BF00115009. URL https://doi.org/10.1007/BF00115009.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.