Reproducibility Report for 'Learning to Play Sequential Games versus Unknown Opponents'

Mahrokh GhoddousiBoroujeni
mahrokh.ghoddousiboroujeni@epfl.ch

Yasaman Haghighi
yasaman.haghighi@epfl.ch

Muhammad Zakwan
muhammad.zakwan@epfl.ch

1 Introduction

This document is the reproducibility report of paper *Learning to Play Sequential Games versus Unknown Opponents*, which considers designing strategies for a learner facing an *unknown* opponent with *multiple types* in a repeated sequential game. The learner, who plays first, chooses a strategy, and the opponent responds to it accordingly. The second player's response function is unknown and can only be inferred by repeatedly playing and observing the responses. Additionally, the learner may face different opponent types at every game round, which he gets to observe at the end of the round. The first player's goal in these games is to find the optimal strategy maximizing a reward function that depends on the unknown opponent's response. Therefore, the learner needs an algorithm to exploit the opponent's response structure, only assessing the observed rewards and opponent type.

To this end, the paper proposes a novel algorithm, *StackleUCB*, which we describe at a high-level below. As a familiar setting of sequential games, the learner maintains a probability distribution over the set of available actions and plays randomly according to it. At every round, the learner then receives a noisy estimate of its' reward and observes the opponent type he faced to subsequently update its' strategy. The strategy update rule is the most critical part of the algorithm. It is built upon the central assumption that similar opponent types and strategies of the learner lead to similar responses. The similarity assumption is conveniently encoded by considering a kernel with known hyperparameters, which is utilized in estimating the opponent's response function via performing standard kernel ridge regression on previously collected data. Once the confidence bound for the opponent's response is estimated through regression, the authors compute the optimistic reward, which could have been obtained from this set of reactions. The updating procedure puts more weight on the actions with higher optimistic rewards. The detailed mathematical formulation can be found in *Algorithm 1*.

The algorithm was then tested on two experiments, for which we provide a short description below.

1.1 Experiment 1

The first experiment depicts a scenario of a network operator(*learner*) who wishes to route some of the 300 available public transportation units between a specified pair of regions in a network of roads, where other users(*opponent*) are traveling according to their preferences. The number of users who intend to travel between each pair of nodes is a random variable and represents different opponent types. The operator wishes to balance the trade-off between the number of routed public vehicles and the overall network congestion through maximizing a *reward* function over successive rounds of the game. Both the operator and the users have a finite and discrete set of actions decoding their origin, destination, and possible routes.

1.2 Experiment 2

The second experiment represent the wildlife conservation task where the goal of park rangers(*learner*) is to protect animals from poachers(*opponent*). The animal density at location y follows a mixture of Gaussian distributions. The ranger aims to protect the animals through maximizing a *reward* function over successive rounds of the game. Similar to *Experiment 1*, both the ranger and the poacher have a finite and discrete set of actions decoding their origin, destination, and possible routes.

33rd Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2 Scope of reproducibility

The principal assertion of the paper is ensuring no regret (c.f. *Theorem 1*), guaranteeing that the learner's performance converges to the optimal one in hindsight (i.e., the idealized scenario in which the types' sequence and opponent's response function are known ahead of time). In addition to this general claim, they present other subsidiary results that we discuss separately for each experiment.

2.1 Experiment 1

The authors compare their proposed algorithm to two extremes when the operator does not route at all or routes all vehicles through the shortest path, leading to the best and worst cases in terms of the average network congestion. They also evaluate their results against two methods proposed in the literature, *Hedge* and *Exp3*. We aim at reproducing all results for this example encompassing the following claims on the average congestion, cumulative rewards, and average regret:

- obtaining average congestion and cumulative rewards as reported in 4,
- comparing the regret of different methods, as done per Fig.1 of the paper,
- establishing the regret converges to zero, which is the main claim of the paper.

The first item shows that the network will have low average congestion, while the second two prove to outperform other algorithms in terms of the eventual regret.

2.2 Experiment 2

This experiment is an example of a single known type opponent game with $\theta_t = \bar{\theta}$. In this problem, $\bar{\theta}$ is revealed to the learner. However, direct reward optimization is not yet possible as $b(., \theta)$ is unknown to the learner. In this regime, the learner uses his previous observations and the optimistic reward function (equation (8) in the paper) to choose the best action. The authors make the following claims:

- Their algorithm is zero-regret. To establish this claim, they compare their algorithm's obtained reward with the optimal, max-min, and best offline models in Figure 2. They show their algorithm reaches the optimal solution after finite play rounds.
- In appendix F, they claim that the general GP-UCB algorithm either converges to a suboptimal solution or needs more iterations to converge to the optimal solution compared to their algorithm.

3 Methodology

In this section, we detail our approach for each experiment. Though the same algorithm is used in both examples, the implementation technicalities and challenges vary widely. For the first experiment, we merely rely on the paper description; while in the latter, we had to access and evaluate the author's Github repository at some point. The reason for this necessity is explained in section 4.2 We re-implement the whole module on regular laptops without involving GPU resources. Apart from the user-developed modules, we take advantage of two Python packages, GPy and NetworkX. The first one develops Gaussian Process and Kernel Learning functions, and the second one accommodates a convenient environment for analyzing networks and graphs.

3.1 Experiment 1

3.1.1 Data and Modeling

The data is based on **?** and equips two parts:

1. Data for a network of regions, modeled as nodes, which are connected through some roads. The graph provides information on the location of the nodes and physical properties of roads, e.g. capacity, length, etc., as well as the network topology.
2. Number of users' travel demands between different pairs of nodes, which is used to construct demand profiles.

On the top of this data, the widely used used Bureau of Public Roads (BPR) model is employed to compute road congestion and travel times in the network **?**. Additionally, the authors model different opponent types by scaling the

demand between each pair of nodes with a uniform random number from $[0, 1]$. Clear explanation for the data and modeling section is given in *Appendix E* of the paper.

3.1.2 Hyperparameters

The most important engaged hyperparameters can be categorised as:

- Operator's action set descriptors: clearly provided in Appendix E.
- Users' preference descriptors: available in Appendix E, though we concluded the paper results are not compatible with this group of hyperparameters and adapted them, as will be discussed in 4.
- Predictor kernel hyperparameters: trained offline via maximum-likelihood according to the given guidelines.
- Random seed: though not mentioned in the paper, we identified the random seed as a critical hyperparameter, that we will set through trial-and-error (c.f. 4).
- Tuning and scaling parameters: either explicitly given in the text, or computed from the theorems. Examples include $\beta, \kappa, \eta, \cdots$.
- Simulation-related: scalars like the time horizon or noise variance, extracted either from the text or the figures.

3.1.3 Experimental setup

We deploy a fully object-oriented implementation approach, with three main classes *Network, Users,* and *Operator*. The *Network* class is built on the top of the data loading procedure, and accommodates an easy representation for the dataset and facilitates graph-related analysis. The *Users* class models how the users react to the strategy selected by the operator, and separates everything on the opponent side from the rest of the modules to make understanding the code easier. Finally, the *Operator* class implements operator's inference and decision making procedures. Each class and the associated attributes and methods are constructed in a separate file, where more implementation details can be found.

3.1.4 Missing Details and Adopted Approach

We gradually obtained very close results to the original paper, as reported in 4. Still, not all details were clear from the text and we adapted our simulations according to precise evaluation of our observations and performing grid-search, where ever necessary. In the rest of this subsection, we detail and motivate our changes in chronological order.

1. *Random Seed:* We soon detected the critical role of fixing a random seed. To account for different opponent types, the authors scale the demand between each pair of nodes by a factor uniformly sampled from $[0, 1]$. The average congestion, and all proceeding criteria, are so sensitive to the number of demands, according to the degree four congestion model described per (17) in the paper. Therefore, for the results of different methods to be comparable, it is crucial to face the same opponents that is achieved via fixing the random seed. It is important not to undermine the stochastic nature of variables by abusing the fixed random seed. Two plausible cases are:
    - Single vs. multiple opponent types: depending on where the random seed is set, we can simulate both single and multiple opponent type scenarios. In the first scheme, the demands are the same at all time steps, which is a strong assumption. Theoretical claims of the paper consider the multiple case, but we should inspect if the provided *values* belong to the single or multiple opponent case, and if *converging to* 0 *regret* happens for both cases.
    - Training and test data separation: kernel hyperparameters are trained offline on a dummy dataset, formed by calculating the average congestion of randomly sampled operator and users action. If the same random seed is used fit training the kernel and simulating the algorithm, it is identical to apriori knowing the opponent type, which conflicts with the main assumptions of the paper.
2. *Expanding Users' Action Set:* First, we focused on routing no public vehicles strategy where the system is derived autonomously only by users' preferences. We observed up to 400% higher average congestion than reported, as well as some intensely over-occupied roads. While the results drastically depend on the selected random seed (standard deviation to mean proportion of approx. 0.6), for all tested seeds the average congestion was higher than paper results. The simple setup of this strategy leaves a low probability of committing bugs, and we concluded that the users' action set in simulations was larger than that explained in the text. We relaxed their preferences by adding up to 5 shortest paths in terms of the physical length to their action set, instead of adding only 3 as written in the paper. We also consider a tolerance factor to model reluctance to taking very long trips. The tolerance factor and number of shortest paths were selected by performing grid-search to reach the closest results to the paper for *no routing* and *always shortest path* strategies.

3. *Probability Update Rule:* Line 6 of StackleUCB updates the operator's mixed strategy proportional to an exponential term of the optimistic reward for each action, $\tilde{\boldsymbol{r}}$, that resulted in an exponent overflow in our experiments. Subtracting a big-enough constant term from optimistic reward of all strategies solves the overflow problem, but produces exponential of negative big magnitude numbers that is numerically bad conditioned. A common solution is to normalize the optimistic rewards to $[0, 1]$ using the minimum and maximum optimistic rewards, $\tilde{\boldsymbol{r}} \leftarrow (\tilde{\boldsymbol{r}} - \min \tilde{\boldsymbol{r}})/(\max \tilde{\boldsymbol{r}} - \min \tilde{\boldsymbol{r}})$. Note that scaling with a constant changes the update rule, counter to subtraction, and therefore, we are not confident if this is the original approach taken by the authors. Additionally, since computing $\min \tilde{\boldsymbol{r}}$ and $\max \tilde{\boldsymbol{r}}$ is computationally expensive, we normalize according to minimum and maximum attained reward on the dummy dataset described before.

3.2 Experiment 2

To examine their claims, we first implemented the OPT, MAX-MIN, Best offline, stackelUCB, and GPUCB models using the paper's information. However, the results were not reproducible, so we used the authors' code to discover their additional assumptions. We will discuss these assumptions later on in the Results section.

The authors model the park area with a 5×5 square divided into 25 sub-squares. We use the center of each small square to refer to that cell.

The animal density at each location of the park is generated as a mixture of Gaussian distributions. Unfortunately, the authors did not indicate the number of Gaussians and the mean and Covariance matrices used to create their dataset. The only information they provided about these distributions is a qualitative representation given in figure 2. We used that figure to estimate the mean and covariances they used, but we could not produce their results exactly. However, this should not affect their claims as their algorithm should converge to the optimal solution after finite play rounds for any density function.

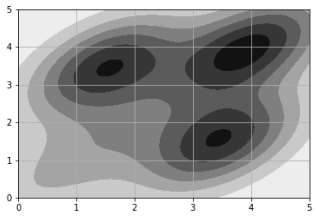


Figure 1: Reproduced density function

3.2.1 Experiment setup

The animals, ranger and poachers could be at any location in the park. As we weren't able to save continuous values, we used a 50×50 grid to divide the park area.

We choose 525 ranger strategies, as stated in the paper. As the first 500 strategies are randomly sampled from the simplex, we could not select the exact set of strategies they used.

We also calculated the ranger's reward, poacher's reward and $b(x)$ based on Appendix F. In addition, we calculated $\bar{\theta}$ as the maximal animal density of each cell.

3.2.2 Model descriptions

- OPT: In this algorithm, the ranger knows $b(., \bar{\theta})$, and it can optimize its' reward directly.
- MAX-MIN: This algorithm maximizes the reward of worse case scenario.
- Best offline: this algorithm models b with a kernel ridge regression model (equation (2) in the paper) using 1000 randomly generated data as the training set. The kernel hyperparameters used in (2) are optimized by Python GPY library using 100 random points, and the kernel type is Matern52. As the process is random, we cannot use the exact data that the authors used, but our final results were similar to theirs.
- Stackle-UCB: We used Algorithm 1 for 100 play rounds. We also did this process ten times for each β_t used in figure 3.

	Shortest route	0% **routed**	**StackelUCB**
Paper average congestion	15.97	1.03	3.51
Obtained average congestion	13.74 ± 0.12	0.80 ± 0.04	5.04 ± 0.38
Obtained *observed* average congestion	13.71 ± 1.12	0.77 ± 1.14	5.19 ± 0.88
Error in average congestion	14%	22%	44%
Paper cumulative reward	21645.4	-813.5	25330.5
Obtained cumulative reward	16258.0 ± 122.7	-807.6 ± 41.9	23744.6 ± 521.0
Obtained *observed* cumulative reward	16290.4 ± 1128.8	-775.2 ± 1147.4	23593.9 ± 1210.0
Error in cumulative reward	25%	1%	6%

Table 1: comparison of results for experiment 1.

- GP-UCB: We use a similar setup as the previous step with 100 play rounds for each β_t. To prove the authors' claim in Appendix F, which says their algorithm converges much faster than GP-UCB, we considered not only the β_t values used in figure 3 but also the ones used in Stackle-UCB to have a fair comparison.

4 Results

We summarize our final results and compare with the results of the paper to evaluate the claims made in section 2.

4.1 Experiment 1

The results for this experiment are summarized in the two panels of Fig.1 in the paper, and are categorized to reports on rewards and on regrets.

4.1.1 Results on Rewards

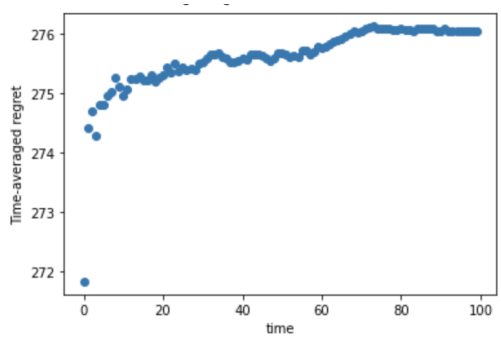
First, we compare and contrast the right subfigure corresponding to the average congestion and cumulative reward with our simulations in Table 1. The values in gray lines are copied from the paper, and the blue lines show our corresponding results. For each value, we provide the confidence bound *mean* $\pm 2\times$ *std*, where mean and standard deviation are calculated with respect to the random seed. Since this is not clear from the text if the authors report the noisy observations or noise-less values that are obscured from the operator, we put two lines encompassing both versions. The difference in the average of the obtained lines is negligible, stemming from the noise being zero-meaned, while the variance of cumulative rewards is essentially different since the noise variances add up. Finally, we compute the relative error between simulations and paper in the purple lines, according to *error* = | *paper - obtained* |*/ paper*, where noise-less obtained values are used.

Before discussing our statistics, we highlight the fact that **the values reported by the authors are inconsistent with each other**. The reward function on page 7 of the paper boils down to *reward = - network congestion* for no routing strategy and to *reward = 300 - network congestion* for always shortest. Thus, the time- averaged and cumulative values in gray lines should be correlated according to the following relationships: *cum. rew. = - 100 av. cong.* for no routing and *cum. rew. = 100 (300 - av. cong.)* for always shortest, where 100 is the simulation time horizon. One can easily see that the aforementioned equations do not hold between the reported values by the authors, and **we critically question this point**. For all values, our obtained results are very close to the results of the paper. We provide a short summary on the sources of mismatch in section 4.1.3.

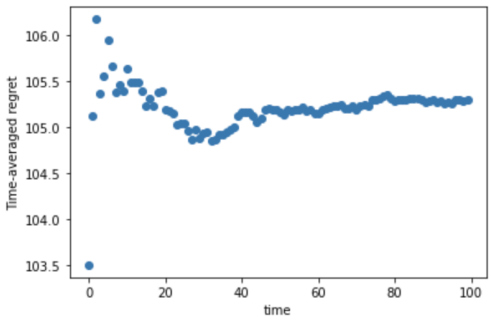
4.1.2 Results on Regrets

The time-averaged regret is plotted in 2. Compared to the results of the paper, we obtained both a lower regret and a faster decrease in it for the StackleUCB method, as plotted in 2c.

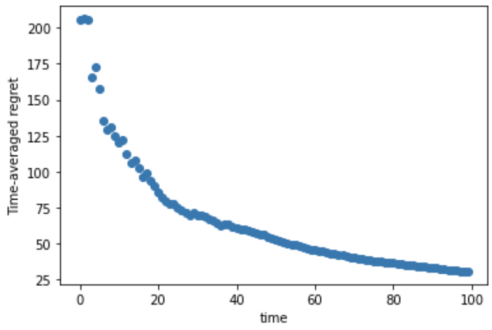
Note that *Fig. 1* of the paper **does not proof the main claim of the paper**, which is converging to 0 regret. They proof a more precise bound for the regret in *Theorem 1*, but they do not compute this bound for the examples. This leaves **no place for evaluating if the main claim of the paper is satisfied** by the eventual regret of around 40. We continued the simulation for a longer time horizon; and observed the decreasing trend. But we once again highlight the faster convergence of our results.



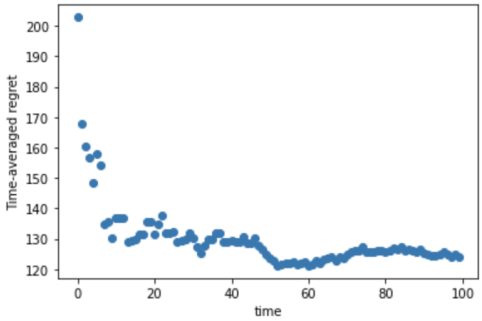
(a) Time-averaged regrets for no routing.



(b) Time-averaged regrets for always shortest.



(c) Time-averaged regrets for StackelUCB with polynomial kernel degree 3.



(d) Time-averaged regrets for exp3.

Figure 2: Comparison of time-averaged regrets for different methods.

4.1.3 Possible Sources of Results Mismatch

Since we detected an inconsistency in the original results of the authors, it is indeed not expected to yield exactly the same results. However, below we describe other reasons that might lead to differences. Two main sources of result mismatch include sensitivity to random variables and unstated or differently treated set of assumptions.

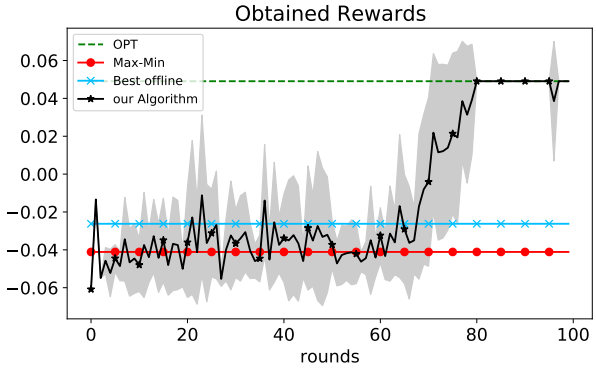
We mentioned the high sensitivity of the average congestion and reward functions before to the randomly scaled demand values before. While we tried to alleviate this dependence by running the algorithm several times and starting from different random seeds, we possibly obtained different demand datasets than the authors. In this case, even changing the arrangement of lines produces different random numbers which leads to substantial differences.

Next, we have made a couple of assumptions which were not present in the report. Most importantly, we changed the strategy update rule due to critical numerical overflows encountered. As argued in section 3.1.4, our changes results in a different probability distribution over actions than the in-implementable version of the paper. We do not know how the authors tackled this problem, and this is another potential source of mismatch. The update rule does not affect the results of *no routing* and *always shortest* strategies, but influences all others heavily.

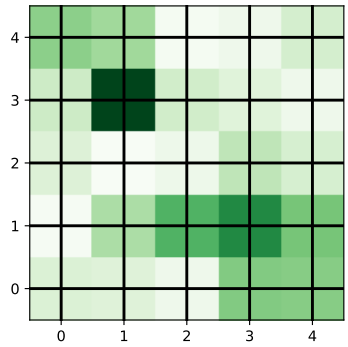
4.2 Experiment2

In our first attempt, the Stackle-UCB algorithm didn't converge to the optimal value . So, we referred to the authors' code and figured out that the poacher reward (18) was calculated with a different formula. They used $R^p(y) = 10 * \phi(y) - \xi \frac{D(y)}{\max_y D(y)}$, but we didn't figure out why the factor 10 was present in their implementation. Also, when we changed this constant to 1 or 100, and their algorithm didn't converge. Hence, we used this new formula for the replication of figure 2.

The results of figure 3 are not precisely the same as the paper, and this could be because of the following reasons: First of all, we didn't use the exact density function as theirs. Secondly, the ranger's strategies are chosen randomly. Also, the ranger observations are noisy. Furthermore, the random data generated for kernel hyperparameter tunning and best offline algorithms are different from the authors'. Finally, we used $\beta_t = 0.01$ to replicate figure 2 because, among all the values used in figure 3, this one had the most similar result to the paper, but there is no guarantee that it

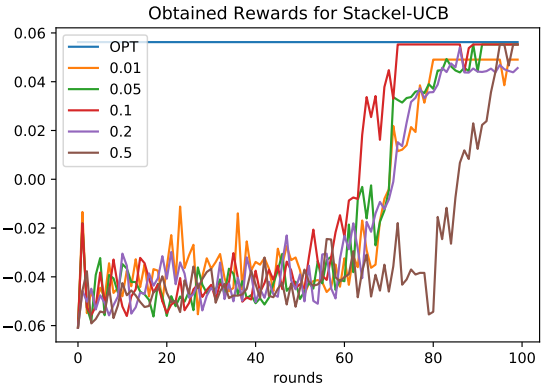


(a) Reproduced obtained reward for different algorithms

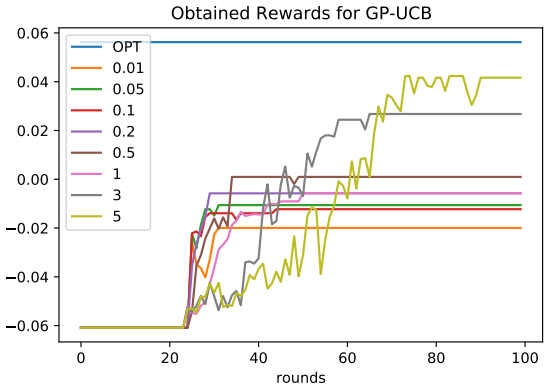


(b) Reproduced ranger's mixed strategy based on paper's formula(8) (probabilities are proportional to the green color intensity)

Figure 3: Paper's figure 2 reproduction



(a) Reproduced Stackle-UCB obtained rewards for different β_t



(b) Reproduced GP-UCB obtained rewards for for different β_t

Figure 4: Stackle-UCB vs. GP-UCB obtained rewards for different β_t

was the same value the authors used. Based on figure 2 replication, we can see that if we use the new $R^p(y)$ formula, we can approximately reproduce their results and confirm that their algorithm is zero-regret. Still, their algorithm seems to converge to the optimal reward much slower than what they claimed, and more than 60 rounds are needed for convergence, though it might be because of the experiment's random nature.

We also replicated figure 3 and we considered the noise standard deviation to be equal to 0.1, which is the value used in Appendix F. The results are in figure 4:

The main reason for the differences between the replicated figure 4 and figure 3 is that the Noise standard deviation used in GP-UCB of authors' code is 0.002. Even though the figure is not replicated thoroughly, we can confirm their claim that their algorithm is faster than GP-UCB and doesn't converge to local optima.

4.3 Summary

For both experiments, we eventually obtained acceptable results. Comparison with all claims in the scope was done in the previous sections, and we highlighted the possible sources of mismatch in!4.1.3. For regret bounds, we obtained a lower bound than the original paper results.

5 Discussion

In this section, we provide a short discussion on our experiment of working on this publication.

5.1 What was easy

The paper is easy to follow and provides well-motivated examples. The discrete action sets in the experiments, simplify the implementations and developing an initial code is straight forward.

5.2 What was difficult

We have discussed several missing details throughout this report, which were for sure difficult to cope with. Apart from these demanding pieces of information, there were a couple of unclear points in the paper(general points are marked by *Gn*, and experiment-specific notes by *E1* and *E2*):

- **Gn)** The regularity assumption in page 3 bounds the reward function to $[0, 1]$, which is not the case for the experiments.
- **Gn)** There is no clear distinction between the noisy and noiseless values. More precisely, the notation of y is used here and there for both versions. This brings up very serious challenges, specially when employing the reward to estimate the opponent's response.
- **E1)** Talking about the noise added to the average congestion, the selected standard deviation of 5 is too big with respect to the range of this quantity. For instance, when contaminating the average congestion of around 3.5 for the StackelUCB algorithm with noise, there is a very high possibility of observing negative values for the naturally-positive average congestion. This brings up the question of how such noise std was selected.
- **E2)** Code was not commented properly and the name of the variables were not meaningful.
- **E2)** The authors used formula for *Eq.(18)* and noise standard deviation for GP-UCB is different from that stated in the paper.
- **E2)** There is no clear instruction on how to generate the animal density.

5.3 Suggestions

A few suggestions from our side to improve the paper are:

- **Gn)** Provide measures on the performance of the regression model, as this might be an important source of slower convergence rate. In our experiments, we conducted tests by performing LOOCV and calculating MSE, R^2, and other statistics calculated by GPy. The performance of our model was very good both on train and validation datasets. This can also be used to compare the effect of kernel order. We observed no substantial differences when simulating with order 3 and 4.
- **E1)** It would be interesting to compare the regret convergence speed when facing multiple opponent types and single type. we provided discussion on how to simulate each case in section 3.1.4.
- **E1)** As mentioned several times, dependence on the demands was one of our greatest challenges. It is possible to alleviate this situation by reporting the proportion of average congestion of different methods with respect to each other, or use a more robust measure.