Application of Imitation Learning to Modeling Driver Behavior in Generalized Environments

Bernard A. Lange[1], William D. Brannon[1]

Abstract— **The need for efficient algorithms for autonomous ground vehicles remains a heavy topic of research interest; this is due to the vast amount of challenges present and the potential benefits it could yield to society. Recent advances in imitation learning algorithms have shown promising results in addressing many of these challenges, but issues regarding accurately imitating emergent traffic behavior have been yet to be steadily addressed. The Stanford Intelligent Systems Laboratory has recently placed efforts into solving this problem through the use of Generative Adversarial Imitation Learning (GAIL) for learning driver policies. Further, they improved upon this algorithm via parameter sharing (PS-GAIL) and reward augmentation (RAIL). Though these algorithms have shown improvements in addressing the problem of accurately imitating human driving behavior through the usage of learned driving policies, it is necessary to validate the overall effectiveness of the learned policies in varying environments. We sought to find the effectiveness of the created driver policies by incorporating changes to the test simulations in two ways: first, we modified the amount of surrounding vehicles that inhabited the original dataset. Second, we changed the roadway setting completely. Following experiments in which the simulations were modified, it was found that the policies learned by the three algorithms performed fairly well when surrounded by a smaller amount of vehicles in the same initial roadway setting. However, none of the three policies performed to standards when placed on an entirely new roadway.**

I. INTRODUCTION

The creation of optimal algorithms in the autonomous vehicle context remains an area of vast research and interest, as numerous complexities in the problem need to be fully addressed for critical matters such as safety of the system and computational efficiency. While testing in necessary in the development of autonomous ground vehicles, simulation is the safest and most economic route in initial development. The need for autonomous driving simulations to effectively mimic situations and traffic scenes that are accurately representative of real situations is crucial in developing driver models; it is necessary to create ways for simulated to vehicles to interact with each other and the roadway environment they inhabit similarly to the way humans do. A current downfall in many modern algorithms ultimately fail to effectively address this, especially in large settings containing interactions between numerous agents. In particular, many learning algorithms fail to properly model the behavior displayed by human drivers; for example, while two separate vehicles may begin from the same initial position and trajectory, their ongoing actions may be drastically different and the difference tends to expand with a growing time horizon. It therefore is necessary to accurately model the emergent behavior of human drivers.

This problem can be effectively modeled as a Partially Observable Markov Decision Process (POMDP), as state transitions can be critical in an average driving scene while the current state may not be known. Further, when multiple agents are under consideration, as in most realistic traffic scenes, the problem can be posed as a Decentralized POMDP, or Dec-POMDP [7]. It is necessary to address the problem as such to find effective solutions. The Inverse Reinforcement Learning approach to Imitation Learning in this project forms the problem as a Markov Decision Process (MDP), in which the states are assumed to be known.

One popular approach to the development of policies meant to mimic human behavior is imitation learning, in which data is gathered via a demonstration (usually of tuple of states with their corresponding actions) and is correlated to an efficient policy [1]. Imitation learning policies are commonly built via Behavioral Cloning (BC), which treats imitation learning as a Supervised Learning problem and a regression model is fit to the state/action space given by the expert. The agent which follows this policy is then theorized to act approximately similar to that of the expert dataset in a testing environment.

While BC policies have been shown to be effective in a general sense, they are largely ineffective in imitating human behavior, such as in driver modeling. This is because BC approaches can only fit well to data explicitly provided by the expert, and tends to perform poorly when in states not explicitly detailed by the expert. In the driving setting, an expert dataset is unlikely to have sufficient data in the spaces that are unlikely to be explored by a human, such as an off-road reaction or a collision with another vehicle. In short, because data is not infinite nor likely to contain information about all possible state/action pairs in a continuous state/action space, BC can display undesirable effects when placed in these unknown or not well-known states. Research has shown that BC policies for driver models usually act undesirably in spaces in which data is not effectively provided, and a cascading effect is observed as the time horizon grows and errors expand upon each other.

Because of problems posed by imitation learning algorithms that develop policies based on Behavioral Cloning, Inverse Reinforcement Learning (IRL) has been explored as an alternative, in which the problem is formulated as an Markov Decision Process and a policy is extracted straight from the data. It is assumed that the expert follows an optimal

*We extend a special thanks to Derek Phillips and Raunak Bhattacharyya of the Stanford Intelligent Systems Laboratory for their constant help and guidance in this project

policy π_E according to a reward function given by

$$R(\pi, r) = \mathbb{E}_\pi \Big[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \Big]$$

discounted by γ [2].

An objective is to solve for the expert reward function via a cost function. The learned reward function, obtained through Inverse Reinforcement Learning, is then used to learn an optimal policy via standard Reinforcement Learning.

Though Inverse Reinforcement Learning has proven effective in learning imitative polices from expert data, its downfall comes in that it tends to be computationally expensive; the Generative Adversarial Imitation Learning (GAIL) algorithm was recently developed by Ho et. al [1] to address this issue. The Stanford Intelligent Systems Laboratory (SISL) applied this algorithm to an autonomous driving environment [2], and created further adaptations in the forms of the Parameter Sharing GAIL (PS-GAIL) algorithm [3] and the Reward Augmented Imitation Learning (RAIL) algorithm [4].

II. Objective

What we seek to contribute is to test the policies learned through the GAIL, PS-GAIL, and RAIL algorithms in an environment foreign to that of which they were learned, with the hypothesis that they would display robustness when facing new situations. This test would effectively indicate robustness if the simulated vehicles driven by the learned policies faced adjustments appropriately; these adjustments are in terms of the amount of vehicles surrounding them and a completely new environment. In addition, we intend to create a tool in which the parameters about the environment can be modified, as opposed to the strict NGSIM environment currently set in place (`https://github.com/sisl/ngsim_env`). When testing a policy in a set unlike that of the training set, it is uncertain if the agent will act desirably; this experiment serves to address whether or not the policies learned from the GAIL, PS-GAIL, and RAIL algorithms are in need of improvement. From a more generalized perspective, this project provides insight regarding direction in further improvements of the current algorithms, and ultimately grants further understanding of policy-learning in imitating human behavior.

III. Background

The GAIL algorithm was created as an appropriate modification to the Inverse Reinforcement Learning approach to policy-learning algorithms to accommodate the fact that it can be quite computationally expensive. Using Generative Adversarial Training, it fits distributions of states and actions given by an expert dataset, and a cost function is learned via Maximum Causal Entropy Inverse Reinforcement Learning[6]. The objective is given by

$$\min_\theta \max_\psi \mathbb{E}_{\pi_E} \log D_\psi(s,a) + \mathbb{E}_{\pi_\theta} \log\, (1 - D_\psi(s,a))$$

in which the learned policy π, parameterized by θ, is passed through. The Wassertein Distance [3] was used to mitigate gradient update problems that were observed. The new objective became

$$\min_\theta \max_\psi \mathbb{E}_{\pi_E}[D_\psi(s,a)] - \mathbb{E}_{\pi_\theta}[D_\psi(s,a)]$$

In this objective, a discriminator function D, parameterized by ψ, outputs a probability that the state-action pair high scores came from the expert policy π_E [1]. It is necessary to learn a policy that is characterized by high entropy via reinforcement learning, in which the differences between the policy state-action occupancy distributions are minimized. This allows for the development of a policy in which the discriminator cannot determine if it is from the expert or not. The parameter θ is updated based on TRPO steps [5], in which the TRPO step serves to prevent the θ value from taking too large of steps due to noise in the policy gradient.

Reinforcement Learning is then applied to develop a policy characterized by high entropy. Kuefler et. al [2] of the SISL applied this algorithm to a driving environment within the Next-Generation Simulation (NGSIM) dataset along a section of the US-101 Highway in California, in which data characterized by real drivers is provided. Simulations were run at a frequency of 10 Hz for a period of 100 steps. GAIL was implemented on this setting with its recurrent neural network structure dependent upon eight core features, displayed in Table 1. In addition to the core features, 16 evenly-spaced measurements of distance from surrounding vehicles are included, along with three indicator features displaying whether or not the vehicle of interest is in an undesirable state (in a collision, off the road, or moving backwards). The measurements from other vehicles are physically indicated by simulated LIDAR beams protruding from the ego agent, as seen in Figure 1. It is noteworthy to mention that the previous actions of the agent are not included in the policy-learning step, which is supportive of the understanding that this problem can be shaped as a POMDP.

TABLE I: Neural Network Core Features

Feature	Units	Description
Speed	$\mathrm{m\,s^{-1}}$	Longitudinal Speed
Vehicle Length	m	Bounding Box Length
Vehicle Width	m	Bounding Box Width
Lane Offset	m	Lateral Centerline Offset
Lane-Relative Heading	rad	Heading Angle in the Frenet Frame
Lane Curvature	$\mathrm{m^{-1}}$	Curvature of Nearest Centerline Point
Marker Distance (L)	m	Lateral Distance to Left Lane Marking
Marker Distance (R)	m	Lateral Distance to Right Lane Marking

It was found that GAIL produced high-performing policies in a single-agent environment [2], where an individual vehicle from the NGSIM dataset was replaced by a vehicle following the GAIL policy. The rest of the vehicles in the simulation were run by their original expert dataset, as specified by NGSIM. However, when the GAIL-policy driven vehicle was placed in a multi-agent setting, in which multiple agents take over the learned policy, this algorithm produced undesirable results among the agents. These results included collisions and off-road driving that are not properly representative of the way humans drive. To mitigate this problem, Parameter Sharing Generative Adversarial Imitation Learning (PS-GAIL) [3]was proposed in which Parameter Sharing Trust Policy Region Optimization (PS-TRPO) [5], an extension of the original TRPO algorithm that allows agents to share a high-performing cooperative multi-agent policy, is applied to the GAIL algorithm. The policy gradient is characterized by

$$\max_{\theta}\ \mathbb{E}_{s\sim\rho_{\theta k},a\sim\pi_{\theta k}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A_{\theta_k}(s,a)\right]$$

$$s.t.\ \mathbb{E}_{s\sim\rho_{\theta k}}[D_{KL}(\pi_{\theta_k}(\cdot|s)||\pi_\theta(\cdot|s)] < \Delta_{KL}$$

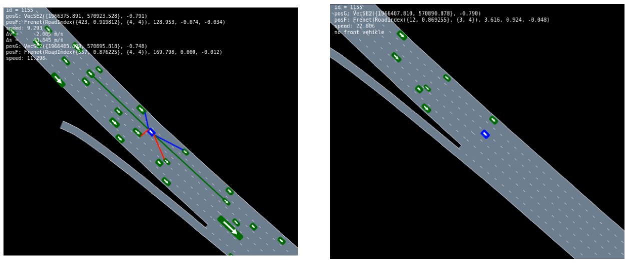
where $\rho_\theta = \rho_{\pi_\theta}$ are discounted state-visitation frequencies induced by π_θ [TRPO]. D_{KL} represents the Kullback-Leibler Divergence between the learned policy distributions in separate optimization steps; Δ_{KL} is a parameter that sets a maximum on the step size of an individual policy change. It was found that PS-GAIL vastly outperformed GAIL in developing effective policies for multi-agent driving models in terms of output such as trajectory deviation, off-road duration, collision rate, and hard brake rate [3].

Though PS-GAIL yielded better results in multi-agent simulations than GAIL, its results still led to undesirable driving characteristics, including unwanted trajectory deviation and off-road duration. To address this, members of the Stanford Intelligent Systems Laboratory introduced reward augmentation to the policy-learning process. The Reward Augmented Imitation Learning (RAIL) algorithm was developed, in which rewards in the form of penalties were applied to undesirable driver activity in the form of binary and smoothed penalties [4]. In binary penalties, a penalty of R is assigned to collision with other vehicles and off-road driving, and a penalty of $R/2$ is assigned to hard braking (acceleration of less than -3 m/s^2). In a smoothed penalty setting, penalties are applied in advance of undesirable actions with the theory that this would prevent these actions from occurring. The performance of the RAIL algorithm was measured in the areas of imitation of local traffic behavior, reduction of local traffic phenomena, and imitation of emergent properties in multi-agent driving.

Of the three algorithms originally tested on the NGSIM dataset (GAIL, PS-GAIL, and RAIL), the RAIL algorithm yielded the highest performance in a multi-agent setting, and it therefore deserves special attention in the implementation of the algorithms in foreign environments.

IV. Challenges

The main challenges that we addressed in approaching this experiment largely involved the NGSIM Environment package developed by the Stanford Intelligent Systems Laboratory. This package is restrictive in that it is set specifically to the NGSIM dataset, the NGSIM roadway (US-101), and the NGSIM vehicle count (while it allows for the shifting of policies between NGSIM vehicles and vehicles created by the imitation learning algorithm, it does not allow for the direct removal of vehicles). Thus, it became an objective to create a tool in which these aspects of the environment could be varied.



```
id = 1155
posG: VecSE2({1966375.891, 570923.528}, -0.791)
posF: Frenet(RoadIndex({423, 0.919812}, {4, 4}), 128.953, -0.074, -0.034)
speed: 9.293
Δv =      2.005 m/s
Δs =     40.845 m/s
posG: VecSE2({1966405.899, 570895.818}, -0.748)
posF: Frenet(RoadIndex({557, 0.876225}, {4, 4}), 169.798, 0.000, -0.012)
speed: 11.298
```

```
id = 1155
posG: VecSE2({1966407.810, 570890.878}, -0.790)
posF: Frenet(RoadIndex({12, 0.869255}, {3, 4}), 3.616, 0.924, -0.048)
speed: 22.806
no front vehicle
```

Fig. 1: (a) Single-agent NGSIM environment depiction of US-101; (b) RAIL policy operating as a single agent

V. Experiments and Results

Our first objective was to recreate the experiments already performed by members of the Stanford Intelligent Systems Laboratory, in which we tested a number of previously-learned policies on the original NGSIM dataset. We first tested GAIL, PS-GAIL, and RAIL policies while leaving the roadway environment the same, in that NGSIM-driven vehicles were initially kept. Following the initial trials, we replaced the NGSIM vehicles with agents driven by our policies. We were able to achieve results similar to those specified in [2][3][4] (see Figure 1a).

Following this, we sought to test the robustness of the policies in a different environment by reducing the amount of vehicles surrounding the ego agent and running the simulation as a single agent. Because the GAIL algorithm, when originally applied to a driving setting, incorporated features into the learned policy that largely revolved around measurements from external vehicles (this feature selection was carried on to PS-GAIL and RAIL), it was theorized that this would provide a measure of performance in a more general sense. In simulating the policies in the NGSIM environment with a reduced amount of surrounding vehicles, it was found that all three policies performed well to a certain extent. The GAIL policy did well in avoiding collisions, but performed poorly in terms of accurately depicting an expert trajectory; it continued to slowly drift right until off the road, and then slowed down immensely. It is necessary to consider that while an indicator function was placed in the policy's neural network structure that signals whether the vehicle goes off-road [2], direct penalties were not incorporated until the RAIL algorithm came into place.

The policy learned through the PS-GAIL algorithm was then tested as a single-agent system in the reduced-vehicle environment. Similar to the GAIL algorithm, it performed well in avoiding outside vehicles, but the speed distribution was much higher for PS-GAIL and it reacted to other agents more similarly to the way typical humans do while driving. However, after passing the NGSIM vehicles, the ego agent sped up immensely and drifted off-road at a high velocity (see Figure 2). The PS-GAIL policy ultimately displayed that within the NGSIM environment it is effective only while surrounded by other vehicles.

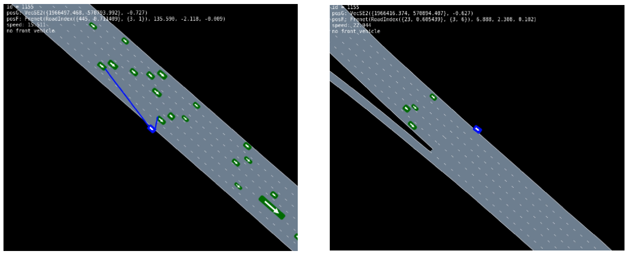


Fig. 2: (a): Single-Agent GAIL driving off the road; (b): PS-GAIL policy-driven vehicle drifting off the road after passing traffic

We continued to test the RAIL algorithm in the same environment conditions. It performed similarly to PS-GAIL when surrounded by external vehicles, and after passing, continued to accelerate. However, unlike with PS-GAIL, the RAIL policy remained stably within the bounds of the road. This can be attributed to the reward augmentation incorporated in the policy learning step of RAIL.

Figure 3 displays the time before going off-road among the different policies within the original NGSIM environment when placed in a single-agent setting. A value of 50 seconds indicates that the policy did not lead the vehicle to go off the road at all. As can be seen, RAIL performed the best, and PS-GAIL's performance increased when the number of surrounding vehicles increased.

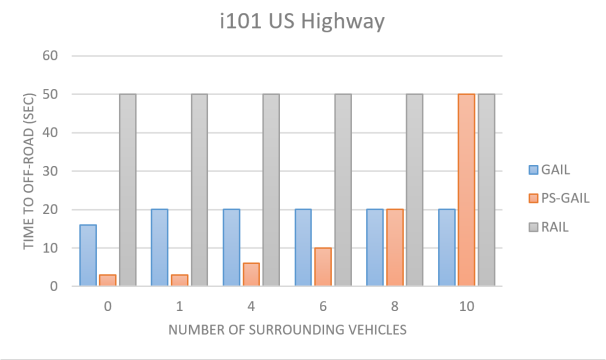


Fig. 3: Time Before Going Off-Road in NGSIM environment of Single-Agent Policy

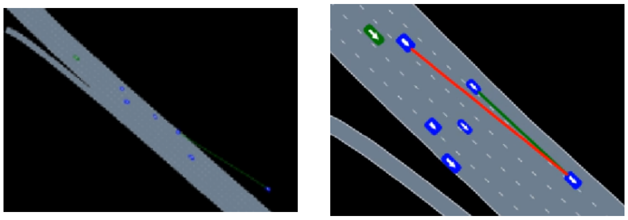


Fig. 4: (a) Multi-Agent PS-GAIL on NGSIM roadway; (b) Multi-Agent RAIL on NGSIM Roadway

Following this, we evaluated the performances of the PS-GAIL and RAIL policies on the NGSIM roadway in a Multi-Agent environment, in which several vehicles were controlled by the learned policies; in our experiments in multi-agent policy testing, we removed all NGSIM vehicles from the scene. Similar to the single-agent environment, the PS-GAIL policy performed well in avoiding collisions and staying on the road only when surrounded by other vehicles. Interestingly, the leader of the group of agents sped forward and went off the road during the simulations while the followers remained together and on the road. RAIL performed as it did in the single-agent environment in that the agents remained on the road and avoided collisions (see Figure 4).

Following the steady removal of external vehicles from the scene, we incorporated the policies into a completely new environment, in which a curvature was introduced. This would allow for a heavier sense of potential performance and introduce complete generalization to the problem. The road used was from the Automotive Driving Models Julia package (see Figure 5 below). The simulation begins in the bottom-most portion of the track, in which the vehicles approach the bottom-right curvature.

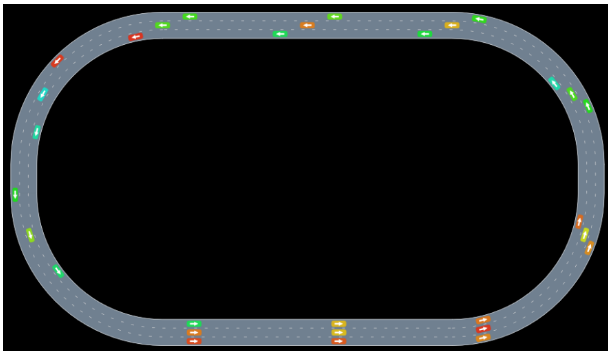


Fig. 5: Oval Roadway Used for Further Testing

On the new roadway, it was found that none of the policies performed to standards, both in single-agent and multi-agent environments; none of them were able to remain on the road to complete the first turn. The GAIL policy traveled extremely slow, and came to a stop when it left the road. The PS-GAIL policy came off the road to its left almost immediately after the simulation began and did not make it to the curvature while on the road. RAIL remained on the road until it came to then proceeded off. As before, the PS-GAIL and RAIL policies accelerated much faster than the GAIL policy, and their turn rate did not compensate nearly enough to appropriately remain on the road during the turn (see figure 6a). The PS-GAIL and RAIL policies also did not perform well when simulated in a multi-agent setting (see Figure 6b).

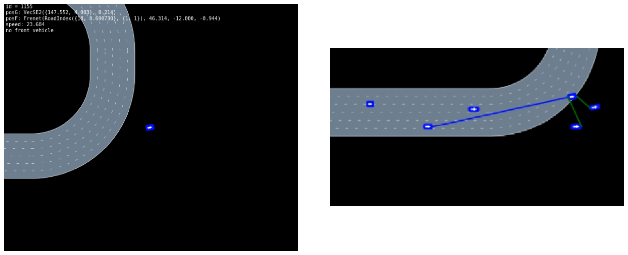

id = 1155
posG: VecSE2({147.552, 4.003}, 0.214)
posF: Frenet(RoadIndex({18, 0.690730}, {1, 1}), 46.314, -12.000, -0.944)
speed: 23.684
no front vehicle

Fig. 6: (a): RAIL Single-Agent Driving Off Road at Curvature; (b): RAIL Multi-Agent Policy Driving Off Road at Curvature

VI. Conclusion

This project served to find the level of functionality that several currently available driving policies, produced by GAIL, PS-GAIL, and RAIL, display when placed in environments outside of that which they were trained. At first, we recreated the experiments previously held by the Stanford Intelligent Systems Laboratory, and proceeded to extend the project to new environments. We did this by first modifying the amount of outside vehicles interacting with the ego agent in both single and multi-agent settings. It was found that all of the algorithms performed well in avoiding other vehicles, but GAIL algorithm tended to drift to the right until off the road. The PS-GAIL and RAIL algorithms accelerated much faster than GAIL, and PS-GAIL led the vehicle off-road when no longer surrounded by other vehicles.

We then proceeded to place vehicles exhibiting the learned policies in a new environment in which a curvature in the road was introduced. It was found that when placed in a setting entirely foreign to the testing environment, the policies did not meet standards. None of them were able to fully accommodate the turn in the road, but GAIL and RAIL did better at detecting the curvature and slightly turned. This observation can likely be attributed to two things: first, heading distance from the end of the road is not included as a core feature in the neural network structure of the original GAIL policy. This is because the policies were originally trained on the straight NGSIM roadway, in which distance to the edge of the road was only considered from the side of the vehicle. Second, specific to RAIL, the smoothed reward augmentation calls for the beginning of penalization when the vehicle is a minimum 0.5 meters from the edge of the road. In this case, this was during a time when the RAIL policy was already driving at a high speed and was therefore unable to stop without hard braking.

In terms of future work, it would be most efficient to improve upon the RAIL algorithm in creating policies as opposed to the other imitation learning algorithms due to its higher levels of performance in both environments. Because domain adaption was presented as the most apparent problem in this experiment, it would be efficient to introduce additional core features to the neural network structure, such as frontal distance from a road edge. This would potentially provide logic on how fundamental changes to the structure of the policy-learning problem via imitation learning could ultimately provide direction in improvement in creating accurate driver models. An additional improvement could be to modify the reward function; it could be made to better account for changes in the roadway setting. For example, it may be beneficial to modify the smoothed reward augmentation to incorporate the need for hard braking when approaching the edge of the road at particular angles; it currently only penalizes hard braking. In addition, continued generalization on the NGSIM Environment package would allow for easier implementation of future policies and variables in testing. It is planned to continue in this project to incorporate some of these improvements, which will grant an enhanced understanding in creating driver models from imitation learning.

References

[1] J. Ho and S. Emron. Generative Adversarial Imitation Learning, 2016.

[2] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. Imitating Driver Behavior with Generative Adversarial Network, 2017.

[3] R. Bhattacharyya, D. Phillips, B. Wulfe, J. Morton, A. Kuefler, M. Kochenderfer. Multi-Agent Imitation Learning for Driving Simulation, 2018.

[4] R. Bhattacharyya, D. Phillips, C. Liu, J. Gupta, K. Driggs-Campbell, M. Kochenderfer. Simulating Emergent Properties of Human Driving Behavior Using Multi-Agent Reward Augmented Imitation Learning, *Under review*.

[5] J. Gupta, M. Egorov, M. Kochenderfer. Cooperative Multi-Agent Control Using Deep Reinforcement Learning. In International Conference on Autonomous Agents and Multiagent Systems, 2017.

[6] B. Ziebart, J. Bagnell, A. Dey. Modeling Interaction via the Principle of Maximum Causal Entropy, 2010.

[7] M. Kochenderfer. Decision Making Under Uncertainty: Theory and Application (MIT Lincoln Laboratory Series), 2015. MIT Press.

VII. Contributions

William Brannon: Became familiar with the programming aspect of the project, but worked on the project from more of a theoretical perspective; read publications to develop background information and gain a firm grasp of the material needed to understand the project. Wrote up most of the results.

Bernard Lange: Did most of the work from a computational perspective; was more familiar with the programming aspect of the project, and implemented the changes in the program. Analyzed the results of the project output.