6-11-2021

Rebalancing shared mobility systems by user incentive scheme via reinforcement learning

Matthew Brian Schofield
Rowan University

REBALANCING SHARED MOBILITY SYSTEMS BY USER INCENTIVE SCHEME VIA REINFORCEMENT LEARNING

by

Matthew Schofield

A Thesis

Submitted to the
Department of Computer Science
College of Science and Mathematics
In partial fulfillment of the requirement
For the degree of
Master of Science in Computer Science
at
Rowan University
May 6, 2021

Thesis Chair: Shen-Shyang Ho, Ph.D.

Committee Members:
Ning Wang, Ph.D.
Anthony Breitzman, Ph.D.

Dedication

I dedicate this manuscript to my loving and supportive family and friends who have made this journey and its continuation possible. My parents, Brian and Wendy Schofield, have always nurtured my academic interests and abilities, for that I thank them.

Acknowledgment

I would like to sincerely thank my thesis committee members Dr. Wang and Dr. Breitzman whose generous guidance and support have not only shaped this work but have shaped my professional abilities.

I would like to particularly thank my thesis committee chair Dr. Shen-Shyang Ho who has guided me through-out my time at Rowan University from my Freshman year until now. It has been an honor to be a part of his lab and I will greatly miss working together on various research projects.

Abstract

Matthew Schofield
REBALANCING SHARED MOBILITY SYSTEMS BY USER INCENTIVE SCHEME VIA REINFORCEMENT LEARNING
2020-2021
Shen-Shyang Ho, Ph.D.
Master of Science in Computer Science

Shared mobility systems regularly suffer from an imbalance of vehicle supply within the system, leading to users being unable to receive service. If such imbalance problems are not mitigated some users will not be serviced. There is an increasing interest in the use of reinforcement learning (RL) techniques for improving the resource supply balance and service level of systems. The goal of these techniques is to produce an effective user incentivization policy scheme to encourage users of a shared mobility system to slightly alter their travel behavior in exchange for a small monetary incentive. These slight changes in user behavior are intended to over time increase the service level of the shared mobility system and improve user experience.

In this thesis, two important questions are explored: (1) What state-action representation should be used to produce an effective user incentive scheme for a shared mobility system? (2) How effective are reinforcement learning-based solutions on the rebalancing problem under varying levels of resource supply, user demand, and budget? Our extensive empirical results based on data-driven simulation show that:

1. A state space with predicted user behavior coupled with a simple action mechanism produces an effective incentive scheme under varying environment scenarios.
2. The reinforcement learning-based incentive mechanisms perform at varying degrees of effectiveness under different environmental scenarios in terms of service level.

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

List of Figures

List of Figures (Continued)

Figure Page

List of Tables

Chapter 1

Introduction

The sub-field of machine learning known as reinforcement learning has rapidly grown in capability and popularity in recent years. In reinforcement learning, agents are trained to learn a policy that they can utilize to perform a task through interacting with an environment based on a performance signal, much like humans. These approaches excel at taking many actions to influence an environment in order to maximize a long-term, often noisy, cumulative reward. Due to this set of strengths, this paradigm has seen great success in the financial markets, robotics, and gaming domains [1] [2] [3] [4]. As the field of reinforcement learning continues to further advance, coupled with the rapid growth of computational power, its use in everyday systems large and small will continue to expand. This field and class of algorithms show tremendous performance potential as long as a particular environment's state, action space, and reward signal can be adequately represented in a way that translates well to the true environment.

A particular area of interest for the use of reinforcement learning is in large-scale shared-mobility systems [5] [6]. With the rapid growth in popularity of shared mobility systems, i.e. car-sharing, bike-sharing, scooter-sharing, as a means of alternative travel options issues surrounding resource management within these systems have presented themselves. Due to the combination of the popularity of these systems and the open-data strategy of their operators, there exists a large amount of data available for analysis towards solving such issues. These systems are based upon a large number of small interactions contributing to overall system performance. This presents an opportune problem setting for reinforcement learning.

1.1 Bikeshare Systems

Shared mobility systems are rapidly growing in popularity in major metropolitan areas. Bikeshare systems operate over a particular geographic area, such as a city or university campus, areas that often have a high volume of both foot and vehicular traffic. These systems allow users to rent a bicycle, or often scooters as well, for a small fee to take short-duration trips. These systems have two popular structures, either they are docked or dock-less. In a docked system there are dock repositories distributed throughout the system's geographic area where bikes can be loaded or unloaded for use. In a dock-less setting, the system's bikes are stand-alone and can be deposited anywhere throughout the system. Both systems also typically employ Global Positioning Systems (GPS) to lock the vehicle if they were to exit the system's geofence.

Over one thousand cities globally have an established bikeshare network. The largest bikeshare network is Hangzhou Public Bicycle in the city of Hangzhou, China launched in October of 2008 with a supply of over 78,000 bicycles currently. One of the most active bikeshare systems in the United States of America is the Capital Bikeshare network with almost 5,000 bicycles and over 3 million users annually. These systems are often operated by a combination of private organizations working with local governments. For example, Motivate LLC acquired by Lyft Inc. operates some of the largest bikeshare systems in the United States, such as the San Francisco Bay Area's Bay Wheels system, Boston's Blue Bikes, Washington D.C.'s Capital Bikeshare, and many more.

The rapid expansion and popularity of bikeshare systems are due to their main benefits in filling a gap in metropolitan commutes, providing an outlet for exercise,

promoting tourism, and they are environmentally friendly. Due to their convenience relative to the traffic and limited parking in urban areas, these systems aid in alleviating the ‘last-mile’ problem as well as serving longer duration commutes across a city. Also, bikeshare systems provide an inexpensive means of exercise and activity for residents. Further, these systems provide a means for tourists to quickly explore various points of interest. The main benefit of these systems is their capacity to alleviate traffic in an environmentally friendly manner that improves the health of users.

A major problem and critique that these systems suffer from are that they regularly become imbalanced and cause clutter in pedestrian pathways. Due to highly correlated user interests and behaviors, vehicles will often be depleted from one area of the system and become congested in another. Resulting in a poor user experience as users cannot reliably secure a bicycle when they are interested in requesting one. For example, in the morning hours, the bicycles in residential areas may all be moved to commercial areas of a city resulting in users in residential areas that arrive later being unable to have their request for a bicycle satisfied. Bikeshare operators often utilize inefficient and expensive methods of maintaining balance, such as using vans or employees to manually move bicycles. Such methods are slow to respond and further add to traffic, which bikeshare systems are intending to alleviate. Capital Bikeshare reported that over half of their operating expenses were tied to rebalancing operations. Recently, methods towards solving this rebalancing problem have become a popular topic in the research community.

1.2 Application of Reinforcement Learning

Reinforcement learning is a paradigm of machine learning that seeks to create an agent that maximizes a reward signal through interacting with an environment [7]. An agent is trained to complete a task or tasks within said environment over a large number of episodes. An episode can be thought of as a game where the agent regularly observes a representation of the environment's current state and takes some action to influence the environment. The agent receives a reward signal after every interaction and seeks to produce the highest possible cumulative long-term reward when the episode ends.

Reinforcement learning has benefitted greatly in recent years with its fusions with deep learning to produce deep reinforcement learning methods, most strongly seen in policy gradient methods. These techniques have produced such achievements as AlphaGo, AlphaStar, and OpenAI Five [2] [3] [4] each respectively learn how to play the games Go, StarCraft 2, and DOTA 2, at a human world champion level. This is an incredible achievement as these games have an effectively infinite number of game states. The latest advancement being forwarded by DeepMind is MuZero, an agent that is able to both learn the rules and surpass a human world champion level ability at classic games such as Chess, Go, and Shogi [8].

Reinforcement learning methods can be utilized to create an agent that can aid in solving the rebalancing problem. A bikeshare system can be represented as an environment through capturing system information, such as its current distribution of supply and past or anticipated user travel interests along with a variety of other environment features. An agent can then interact with the system through offering users incentives that slightly alter their behavior. By slightly influencing the behavior of

individual users, large changes in the operation of the system can be achieved. This agent can receive a reward signal in the form of the ratio of the number of users who requested a bicycle that were able to receive one, this captures an improvement in the overall user experience and system functionality. The agent can also be further constrained by a budget that it must learn to strategically utilize to maximize the system's level of service.

Bicycle supply balance is an important element for a bikeshare system's operation, however the main concern for bikeshare system operation is maximizing the number of users that are able to receive service within the confines of a given budget. Likewise, what should be optimized is not a representation of the equal distribution of bikes throughout the system, but instead the number of successfully serviced user. While an equal supply distribution is not directly being optimized it is likely to still be produced as to offer the highest servicing opportunity for users. Further, if a high service level for users can be maintained the supply of bicycles within the system can be reduced. Lastly, a high service level will improve the user experience of the system and it can be reasonably speculated that this will grow the popularity of system, something operators would be very interested in.

It is critical to analyze the effectiveness of varying state and action representations as well as the impacts of varying levels of system constraints and environment parameters. Many current approaches have a very large state space or a very difficult to utilize action mechanism, analyzing how these representations can be tuned can lead to much greater performance. The impact of varying budgets, user demands and interests, and supply distributions, is a critical component in understanding the comparative performance of these various representations.

1.3 Scope and Contribution

Methods for improving the service level of a bikeshare system can be further extended to apply to other shared mobility networks as well as general resource networks. Ride-sharing services can benefit from these findings to implement similar incentive structures to promote a wider availability for their service in particular areas to improve overall user experience. While they may see a reduction in profit for individual rides through utilizing incentivization, they are likely to see an increase in positive user experience and brand reputation that can lead to larger volumes. Larger generalized resource networks can use incentives to improve resource availability in areas of a system that would otherwise have requests unfulfilled.

In this work I compare the effectiveness of various structures of reinforcement learning agents and representations as well as their performance over varying bikeshare system environmental constraints and representations. Synthetic data used in experimentation and analysis is derived from real user trip data made publicly available by Washington D.C.'s Capital Bikeshare system. Key contributions of this work are:

- Empirical results show that space representations of shared mobility systems can be improved through a reduction in dimensionality and through the incorporation of predictive models.
- Empirical results show that action mechanisms in a shared mobility system's reinforcement learning framework can be improved by incorporating incentives for users to move specifically in each direction rather than in any direction arbitrarily.

- An in-depth analysis showing the impact of varying environmental constraints such as the level of supply, user demand, and available budget.

Chapter 2

Literature Review

This literature review seeks to identify the current standards and past paths in both reinforcement learning research and computational techniques for the bikeshare rebalancing problem.

2.1 Bikeshare Rebalancing

Bikeshare systems are a relatively popular area of research interest due to the focus on open data by their operators. This access to real world data of crowd behaviors is enticing to many researchers and is beneficial to bikeshare system operators who can receive large amounts of analysis. As of April of 2021, google scholar has indexed roughly 260 academic works focused specifically on bikeshare system rebalancing and optimization.

One of the earliest works on learning a dynamic user incentive pricing strategy to promote balancing in a Bikeshare system is Pfrommer et. al. [9]. They propose an incentivization mechanism that encourages users to slightly modify their destination behavior to position bikes near underfilled stations as to reduce manual repositioning costs. Additionally, they propose that during rush hours a truck routing scheme for added redistribution capabilities. Further, they model their simulations based on historical London's Barclays Cycle and they compare their algorithms performance of payouts to users versus the cost of hiring repositioning staff.

The first case of an online learning system to generate an optimal pricing policy deployed into a real system is Singla et al. [10]. They call their algorithm DBP-UCB

Dynamic Budgeted Procurement using Upper Confidence Bounds, with the idea that the learning of how to attribute a budget *B* across *N* users can be decoupled from the specific parameter settings of *B* and *N*. Their system learns through regret minimization how to best propose alternative pick up and drop off locations in exchange for a monetary incentive. This system was beta tested in a European city by the bikeshare operator MVGmeinRad and showed success over a 30-day period versus a truck based rebalancing approach.

Ghosh et al. [11] proposed an optimization model that would assign re-positioning tasks to users. They created a model of this problem setting known as Dynamic Repositioning and Routing Problem using Trailers (DRRPT) where users would utilize trailers to rebalance bicycles and they would compete within an auction to bid for repositioning tasks. Their algorithm relies on the Vickrey-Clarke-Groves (VCG) mechanism to assign tasks to bidders.

Lv et. al. [12] proposed a crowd sourcing approach through an auction method Truthful Predicted Task revenue TruPreTar. Under a budget constraint, users are able to bid for repositioning tasks, and then the system will determine rebalancing task allocation and payments. They utilize a bipartite graph matching technique, where a graph *G* maintains users and tasks bid on. Then in a sub-graph of *G* known as *G'* a pairing from tasks to users is satisfied where high-value tasks and low-cost users are prioritized.

Chahchoub et al. [13] utilize an outlier detection methodology to identify stations that are either nearly empty or full based on a simple occupancy rate calculation. Outliers are determined based on Gower's similarity degree and a Moran scatterplot that isolate

stations that are outliers relative to other nearby stations in their ‘neighborhood’. Then users are suggested slightly altered routes that will help to rectify these outliers.

Chiariotti et al. [14] consider the rebalancing and incentive problem as a joint optimization problem given the current state, arrival, and departure rate. A three-step procedure is used to approximate the solution and compute the new state, adjusted arrival rate, and adjusted departure rate. The approach requires users to pick up or drop off their bikes to enable rebalancing.

Li et al. [15] proposed two algorithms that within a static setting seek to maximize the number of the served users and minimize their trip time. They formulated this optimization problem as the weighted k-set packing problem. They designed a Greedy Trip Planning algorithm (GTP) and a Humble Trip Planning algorithm (HTP). They compare their two proposed algorithms versus a Random Trip Planning algorithm (RTP) and report to show that both GTP and HTP outperform RTP. They analyze the various environmental conditions that impact the performance of their various algorithms.

Tomaras et al. [16] proposed the algorithm MultimOdal Trip Rebalancing (MOToR) which seeks to combine previous approaches of predicting demand and applying a-posteriori rebalancing methods. They utilize the OpenTripPlanner framework, a popular collection of open-source projects that coordinate analysis of interconnecting transportation networks. They incorporate real-world travel delays into their simulations and attempt to optimize for system supply balance.

An et al. [5] propose an actor-critic reinforcement learning approach to learn a rewarding mechanism (picking up/parking bonus) for a car-sharing system and allows

continuous action space. The reward mechanism is used to guide the users' behaviors through price leverage to ensure cars are parked in areas in need of supply and prioritize picking up where cars are in a high supply, boosting the company's profit and service level.

Pan et al. [6] have proposed a hierarchical reinforcement learning algorithm known as Hierarchical Reinforcement Pricing (HRP) to learn a policy using the deep deterministic policy gradient (DDPG) algorithm to incentivize users under a budget constraint to optimize the system's service level. A static incentive is applied to each region at the start of each time slot that is used to incentivize users to move to a neighboring region if they would otherwise be unable to begin their trip. A key point of their hierarchical approach is that each region computes a local Q value based on the action's performance and then a summation of these Q values is used to train the Critic and by extension the Actor networks of HRP's DDPG component. They primarily utilize the metric service level, rather than system balance.

Duan et al. [17] expand upon Pan et al.'s framework to include an incentive to alter the destination selection behavior of users to further improve a system's long-term performance. They propose utilizing a separate budget allocation for the source interest deviation incentives and the destination interest deviation incentives. Further, they introduce a maximum detour distance constraint calculation, such that if this constraint is not satisfied the user would reject the deviation incentive.

2.2 Reinforcement Learning

This section of the literature review provides background regarding reinforcement learning, particularly its evolution towards today's deep reinforcement learning. According to Sutton and Barto [18] reinforcement learning's early history involves two separate threads in animal psychology research and optimal control in system dynamics research. Today reinforcement learning research has combined these threads and is primarily focused on training deep neural networks as underlying policies.

In the 1950s and 1960s, Richard Bellman laid much of the groundwork for reinforcement learning through his work in optimal system control [19] [20]. He introduced concepts that evolved to today be known as dynamic programming, Markov Decision Processes (MDPS), and the Bellman Equation. Later work in the 1970s through 1990s focused on dynamic programming and introduced further concepts such as partially observable MDPs, approximations, and asynchronous search that evolved into reinforcement learning. In combination with psychology research that described the concept of trial-and-error search and the "Law of Effect" [21], where it was observed that positive and negative reinforcement in connection with events leads to changes in the behavior of animals [18].

One popular branch of modern reinforcement learning is Q-learning, originally introduced by Chris Watkins in his Ph. D. thesis as a means to solve delayed reward tasks in MDPs. The idea of Q learning is to estimate a value through trial-and-error for each action that can be taken in a state that correlates to that action's long-term reward. Mnih et al. [22] proposed the Deep Q Network (DQN) algorithm which trains a neural network to estimate the Q-values for each action given the environment's current state as an input.

Silver et al. [23] expanded on the DQN algorithm to introduce the Deterministic Policy Gradient algorithm (DPG) which directly predicts an action, allowing for continuous actions, rather than DQNs limitation to only produce a probability distribution over a discrete set of actions.

Recent developments in Q-learning are focused on the use of the Actor-Critic architecture [24] [25]. Silver et al. [26] proposed the Deep Deterministic Policy Gradient (DDPG) algorithm where one neural network, the Actor, predicts an action based on an input state and a second neural network, the Critic, estimates the Q-value for the current state and the predicted action. This provides added stability over the single network used in the DPG algorithm. Mnih et al. [27] proposed Advantage Actor-Critic (A2C) where there is one critic network that learns from multiple actor networks that work in parallel and sync every iteration.

Schulman et al. [28] introduce Trust Region Policy Optimization (TRPO). TRPO is proposed as a policy gradient method with a focus on reliable monotonic improvement within a large neural network policy. TRPO utilizes an approximation of the Kullback-Leibler divergence between the current policy and a policy update as a constraint to ensure that a new policy is similar in performance. This is to avoid large shifts in a policy that can damage performance. Empirically TRPO achieves steady performance improvements while not suffering from the instability seen in other methods.

Wu et al. [29] introduce Actor Critic using Kronecker-Factored Trust Region (ACKTR). This seeks to improve on TRPO by utilizing a Kronecker-factored approximation in its gradient calculation to improve sample efficiency and reduce

computational complexity. Kronecker-Factored updates can update the network layer-by-layer rather than updating all layers at once, making it much faster to update a large dense neural network policy.

Schulman et al. [30] expand upon their TRPO algorithm and propose the use of Proximal Policy Optimization (PPO) Algorithms. They claim that PPO maintains TRPO's reliable performance, but has a simplified implementation, improved sample complexity, and they suggest it is more robust in requiring less hyperparameter tuning. PPO introduces a clipped objective function that bounds a policy's change in performance on updates replacing TRPO's KL Divergence constraint.

Chapter 3

Background

In this chapter, I will discuss the typical framework of reinforcement learning as well as key concepts, trade-offs, and types of algorithms. Machine learning is often viewed as having two paradigms, supervised and unsupervised learning. However, reinforcement learning presents itself as a third lesser-known paradigm [18]. In supervised learning, an expert provides a set of labeled objects to train a model in the hopes of the model generalizing to successfully label similar objects outside of its provided training data. In unsupervised learning, an algorithm is used to identify structure and associations in unlabeled data. reinforcement learning utilizes techniques and concepts from both paradigms, as well as applying its own unique techniques. reinforcement learning relies on a well-designed simulation environment which can be seen as analogous to an expert labeled input in supervised learning [18]. However, much of reinforcement learning is self-guided exploration with only noisy reward feedback which can be seen as similar to unsupervised learning.

3.1 Reinforcement Learning Framework

Reinforcement learning is often formalized using a Markov Decision Process (MDP), a classic method of formalizing sequential decision making. An MDP is a 4-length tuple that described a process comprised of the process's state space, action space, transition probabilities, and transition rewards, or (S, A, P_a, R_a). The state space is the set of all possible states that may be produced by an environment as represented to an observer. An MDP's state space can be designed to highlight high-level information

about an environment, or it can present a very high-dimensional representation whose structure must also be learned by an actor attempting to learn to perform within the MDP. For example, a state-space for the Atari game Pong may represent the environment as the x and y coordinate locations of the two players and the ball, or the environment may be represented directly by a screenshot of its 160x192 pixel screen.

An MDP's action space defines how an actor is able to interact with the environment. For example, the action space in the aforementioned game Pong would be moving the player's paddle vertically up or down. An MDP's transition probabilities P_a maps the dynamics of changing from the current environment state, *s*, to a new state, *s'*, based on the action, *a*, taken by the actor, defined as P(s' | s=s, a=a). For example, in the game blackjack at any particular game state a player may take the hit action which has varying probabilities of moving the current state to a new state as represented by the cards in the player's possession and remaining in the deck. Lastly, the reward function R_a(s, s') outputs the reward after taking an action *a* in state *s* and arriving in state *s'*. An MDP's dynamics can be summarized by Equation 1. The goal in 'solving' an MDP is to create a policy π(s) that maps the current state, s, to the action that will maximize the cumulative long-term reward received.

$$p(s', r \mid s, a) = Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\} \quad (1)$$

Figure 1

Example Markov Decision Process for the Game of Tic-Tac-Toe

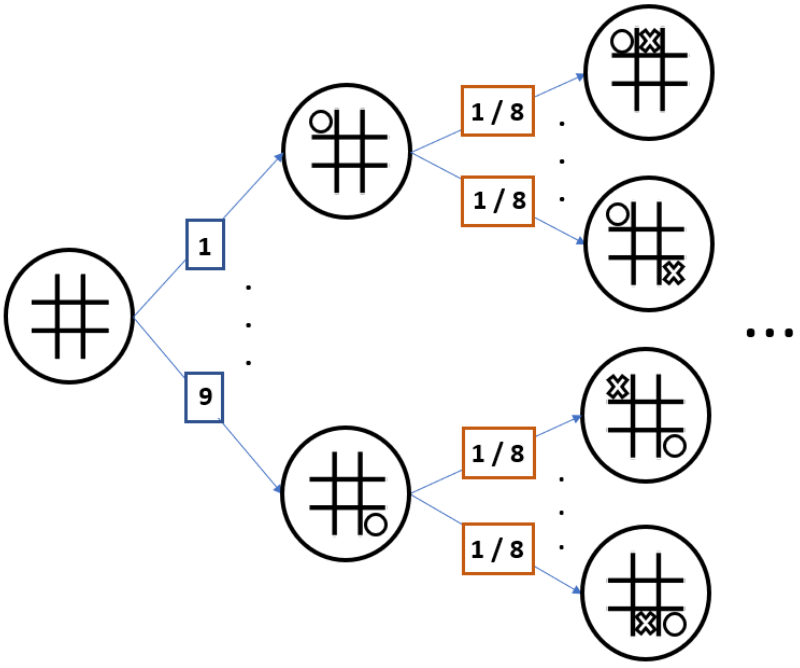


Figure 1 shows an example diagram of an MDP in the context of a tic-tac-toe game versus a random opponent agent. The game begins with a static initial state, S_0, an empty board shown furthest to the left and the player is presented with the action space A={1,…, 9} where the player's 'O' is to be placed. The random agent then has a 1/8 probability to place an 'X' in any of the remaining eight squares. Even in this simple game of tic-tac-toe, its MDP allows for over a quarter million possible games. Likewise, in more complex environments with higher dimensional state and action spaces, and probabilities state transitions there can be a greater number of possible trajectories than atoms in the universe [4].

3.2 Key Concepts in Reinforcement Learning

A key concept in reinforcement learning is the trade-off between exploration and exploitation. This is in reference to the dilemma that in order to obtain a high cumulative reward the agent must exploit their policy which represents their current best understanding to achieve a high reward. However, in order to discover new and possibly more lucrative opportunities an agent must explore new states and actions. A general standard has been adopted that agents are designed to have a strong initial exploration bias that then decays over time towards agents arriving at a strong exploitative bias. Conceptually this can be seen as when first introduced to a new environment experimenting with its various inputs to form an understanding of its outputs. Then as the general mechanisms are understood, utilizing current knowledge to arrive at a more distant state and occasionally attempting new inputs to observe if a better path can be found. In reinforcement learning the probability that an agent will take an explorative action is denoted as epsilon, ϵ, and as the agent continues to learn in the environment ϵ is decayed often to a fixed minimum. An interesting side effect of this is that an optimal agent's average return will be bounded to the optimal return minus the fixed minimum ϵ, as there is a minimum ϵ chance at any timestep that even an optimal agent will take a non-optimal action [18].

Another core concept in reinforcement learning is the idea of delayed reward. The overall goal of reinforcement learning is to maximize the cumulative reward from interacting within an MDP by maximizing the Equation 2. However, this does fail to recognize infinite MDPs and the concept of near-term rewards often being superior to future rewards. A discount rate denoted as gamma, γ, is utilized to apply a discount to

future rewards as shown in Equation 3. γ is in the range [0, 1), where 0 indicates the agent is only interested in immediate reward and values close to 1 indicate that the agent is just as interested in long term reward as they are immediate. Typically, in practice γ is set in the range [0.9, 1.0), providing a strong incentive for long-term reward.

$$G_t = \sum_{t=1}^{T} R_t \tag{2}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3}$$

Value functions are popular components of reinforcement learning algorithms, these seek to quantify the value of a given state, the expected future reward from a given state if the current policy is followed. In Equation 4 we can see the definition of the value function, where given the current state, *s*, the expected cumulative reward considering taking all following actions, *a*, according to the current policy, π. This is often used when an agent is planning ahead, given the option to choose between a set of states the agent can estimate a comparative value for each. There also exists a theoretical variant of the value function $V^{\pi}(s)$ that is $V^{*}(s)$ where * denotes the theoretical optimal policy, thus $V^{*}(s)$ would return the ground-truth value of a state.

$$V^{\pi}(s) = E_{a\sim\pi}[G_t|S_t = s] \tag{4}$$

The Q function, also known as the action-value function, is to quantify the benefit of taking a particular action in a particular state. Many reinforcement learning algorithms seek to calculate or predict Q values for each possible action given the current state, then take the action that is associated with the highest Q value. In Equation 5 the definition of

the action-value function is described, where given the action, *a*, in state, *s*, the cumulative reward is calculated if then the current policy, π, is followed. Similar to the optimal value function there also exists an equivalent optimal action-value function in $Q^*(s,a)$.

$$Q^{\pi}(s,a) = E_{a\sim\pi}[G_t|S_t = s, A_t = a] \quad (5)$$

A popular combination of both value and action-value functions is the advantage function, A(s,a). This is defined in Equation 6 and quantifies the advantage of taking an action in a given state versus following the action that would be prescribed by the current policy π. This is often used in comparing the effects of a new policy relative to the current policy.

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \quad (6)$$

3.3 Policy-Gradient Paradigm

One of the major paradigms of reinforcement learning is Policy-Gradient (PG), in experimentation I will primarily focus on this paradigm due to its latest advancements [29] [28] [30]. PG algorithms are on-policy approaches that seek to optimize a policy directly as it interacts with the target environment. A trade-off of on-policy approaches is that they cannot reuse old experiences, however the experiences that these approaches do learn from are very relevant to the agent. PG approaches are designed to increase the probability of high reward actions and decrease the probability of low reward actions for each given state. These methods are also known as model-free as they do not need access to a complete model of the environment they are interacting with.

The core objective of PG approaches is to optimize a parameterized policy π_θ, a deep neural network, through its parameters θ using the gradient $\nabla_\theta J(\pi_\theta)$. The Vanilla Policy Gradient (VPG) is a foundational approach to policy gradient. Episodes collected through interaction with an environment are used to optimize an agent's underlying policy. Gradient updates, as denoted by $\nabla_\theta J(\pi_\theta)$, are applied to the current policy parameters θ_k given a learning rate *a* to produce the next iteration of the policy parameters θ_{k+1} according to Equation 7. Equation 8 shows the gradient update calculation after an episode. The term $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is the gradient of the log probability or change in probability of arriving at the action a_t given the input environment state s_t. Further, $A^{\pi_\theta}(s,a)$ is the advantage function which denotes the 'advantage' of taking action *a* in state *s* over following the action prescribed by the current policy π_θ. Equation 8 produces a gradient that increases the probability of taking actions that are an improvement over the current policy's action choice given state *s* and decreases the probability of taking actions that reduce performance.

$$\theta_{k+1} = \theta_k + a\nabla_\theta J(\pi_\theta)|_{\theta_k} \tag{7}$$

$$\nabla_\theta J(\pi_\theta) = E_{e\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s,a)\right] \tag{8}$$

One of the latest advancements in policy gradient approaches is the Trust region Policy Optimization algorithm (TRPO) [28]. Many reinforcement learning algorithms suffer from instability due to parameter updates that greatly change policy behavior. TRPO seeks to address this issue by applying a trust region constraint in order to keep policy updates close in performance rather than simply close in parameter space. This

trust region constraint is really a KL-divergence constraint on the change in action probability distributions. An update to the policy will only be allowed if the following constraint is satisfied for hyper-parameter δ:

$$D_{KL}(\theta||\theta_k) \leq \delta \tag{9}$$

Further, rather than considering the change in log probabilities of actions TRPO considers how a proposed policy performs compared to the previous policy iteration through the surrogate advantage function. Shown in Equation 10, the probability of selecting an action, *a*, given state, *s*, between the proposed policy θ and the old policy θ_k is evaluated using old data. If the $L(\theta, \theta_k)$ is positive and the new policy θ satisfies the KL divergence constraint shown in Equation 9, then the policy is updated to the proposed new policy θ.

$$L(\theta, \theta_k) = E_{s,a\sim\pi_\theta}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a)\right] \tag{10}$$

Following TRPO the algorithm Proximal Policy Optimization (PPO) was proposed as a way to reduce the implementation complexity, reduce the computational complexity, and improve the sample efficiency of TRPO through the use of a clipped objective function. PPO utilizes mini-batches of Stochastic Gradient Descent to propose a new policy θ. Then rather than seeking to satisfy the KL Divergence constraint, the clipped objective function shown in Equation 11 is used. This equation is computationally much more efficient and is simpler to implement. The first argument to the min operator is intended to undo an update that results in a disadvantages action becoming more likely, as this will only be the min of the two arguments when the

advantage function $A^{\pi_{\theta_k}}(s,a)$ is negative. The output of clipping function will otherwise be utilized in an update. The purpose of clipping is to bound the impact of an update even if it is seemingly very good, do to $A^{\pi_{\theta_k}}(s,a)$ being an unreliable estimator in practice as it is approximated by a neural network itself.

$$E_{s,a\sim\pi_\theta}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a), clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\ \epsilon, 1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right)\right] \qquad (11)$$

Chapter 4

Problem Setting

In this chapter, the problem setting will be fully defined as to better understand following discussions around data simulation, experimentation, and analysis. This chapter details how the constructed problem circumstance maps to the modeled real-world setting.

4.1 Environment

The environment models a shared mobility system, specifically a bikeshare system, as a 2-D *n x m* matrix spatial representation. This matrix, *R*, represents each region in the system $R = \{r_{11}, r_{12}, \ldots, r_{nm}\}$. Each element *r* represents the features of each region giving the matrix R a channel depth determined by the region features included in a representation's implementation, such as supply, user arrivals, and user destinations. The activity within a shared mobility system is discretized into T time slots, e.g. time slots $\{t_0, t_{1, \ldots,} t_T\}$. In experimentation, an hourly-based system is utilized where T=24 is utilized to represent the 24 hours of a day. Before the simulation begins at t_0 the system will be initialized with a supply represented as *S* of vehicles distributed according to a given statistical distribution across regions. Every time slot the supply of resources available in each region will change due to dynamics in user arrivals in the current time slot and destination patterns from trips that began in the previous time slot.

Each day *N* users will appear in the system in need of service distributed amongst each time slot according to an arbitrary hourly user activity distribution. When a user appears, they will have an intended arrival location r_{ij}^a. If there exists a resource in r_{ij}^a the

user will occupy that resource, removing it from the region, and begin their trip to their intended destination region r_{ij}^{d}. However, if the user appears in r_{ij}^{a} and there exists no available resource, an incentive can be offered to the user such that they may be incentivized to move to a neighboring region that does have an available resource. If successfully incentivized, this neighboring region is then the user's new arrival region and they will begin their trip to their intended destination region r_{ij}^{d}. Also, note that in some extended implementations the user can also always receive an offer to similarly alter their destination region. The neighboring regions available for a user to move to from their current region, r_{ij}^{a}, is defined by the neighbor region vector:

$$N(r_{ij}) = (r_{i+1,j}, r_{i-1,j}, r_{i,j+1}, r_{i,j-1}) \tag{12}$$

The dynamics of a user's trip is as follows. When a user's request is satisfied and the user chooses to occupy a resource, that resource is removed from its current region and placed into a buffer known as the "Travel Buffer". At the end of the current time slot, after all users have either begun a trip or have left the system having failed to be serviced, this buffer will resolve to allocate all resources stored within to their destination region ready to be used in the following time slot. This buffered approach is to simulate the time requirement of users traveling with a resource.

4.2 User Model

In this environment, an agent can offer a user an incentive to slightly alter their behavior, through altering the region in which they select a resource or in some instances alter where they deposit the resource. The agent can supply an offer vector with offers o_{pq}^{t} at time slot t for all neighboring regions, denoted r_{pq}, in the neighbor set $N(r_{ij})$ for the

user's arrival region r_{ij}. An offer o_{pq}^{t} must be in the range [0, I_{max}], where I_{max} is set by the environment as the maximum incentive allowed to be offered to a user. This I_{max} limitation is imposed as if implemented in a wider system there would a set maximum possible incentive as to avoid exploitation. Offers also cannot be negative as the user would simply not accept such an offer and may execute that action outside of the system regardless.

Given an offer, a user will be modeled as considering an offer through the user cost model shown in Equation 13, described by Pan et al. [6]. In Equation 13, d is the Euclidean walking distance from the user's location in region r_{ij} to the closest resource in region r_{pq} and η is a positive weight to account for currency adjustment. In this environment a resource in the same region as the user costs nothing, 0, to move to. Further, if the resource's region is not immediately adjacent to the user's region i.e., r_{pq} is not in the neighbors vector for r_{ij}, then it is simulated that the user would not consider moving to it and their cost is represented as ∞.

$$cost(r_{ij}, r_{pq}, d) = \begin{cases} 0, r_{pq} = r_{ij} \\ \eta d^2, r_{pq} \epsilon\, N(r_{ij}) \\ +\infty, otherwise \end{cases} \tag{13}$$

Once both an offer and the cost to accept said offer has been established for a user, then the utility of the offer to the user will be calculated according to Equation 14. The incentive utility u_{pq} determines how beneficial to a user a given offer to move from region r_{ij} to region r_{pq} for an available resource is. If u_{pq} is positive, then accepting the offer is a net benefit to the user and they would be willing to follow through. If u_{pq} is negative, then accepting the offer would not be beneficial to the user and the user would

not accept the associated offer. Lastly if u_{pq} is 0, then the offer's utility to the user is net neutral and the user would be willing to follow through with the associated offer.

$$u_{pq} = o_{pq}^{t} - cost(r_{ij}, r_{pq}, d) \quad (14)$$

A user's decision regarding a given offer vector is determined by the circumstances and equations defined above coupled with the following final determination logic. A user will be simulated to consider the utilities for each offer and neighbor region constructed as a vector described in Equation 15. If argmax(û) >= 0, then the user will move to the associated region r_{pq} that corresponds to the maximum u_{pq} and their request will be satisfied. Otherwise, if argmax(û) is negative then the user's request will not be serviced.

$$\hat{u} = u_{pq}, \forall r_{pq} \epsilon N(r_{ij}), where\ r_{pq}\ supply > 0 \quad (15)$$

4.3 Objective

The primary objective in this problem setting is for an agent to maximize the number of satisfied user requests, or as a ratio defined as service level according to Equation 16. Additionally, there exists a budget constraint *B* where every time a user accepts an offer that offer amount is deducted from the budget *B*. If the offer is greater than the remaining budget *B*, $o_{pq}^{t} > B$, the offer cannot be carried out and the user's request will not be satisfied. Using reinforcement learning a policy can be constructed that is able to effectively utilize a given budget *B* to improve the service level performance within a shared mobility system. Through analysis and experimentation, various reinforcement learning representations will be evaluated regarding their performance towards this objective.

$$Service\ Level = SL = \frac{fulfilled\ user\ requests}{total\ user\ requests} \quad (16)$$

4.4 One-Dimensional Scenario

Prior to conducting experimentation in the previously detailed problem setting a simplified one-dimensional scenario was explored for familiarization with the problem domain. In this section, I will detail the one-dimensional problem setting and distributions used in simulation [31].

4.4.1 Problem Setting

In the one-dimensional setting the system is represented as a vector of length *n* R= $\{r_1, r_2, \ldots, r_n\}$, rather than as an *n* x *m* matrix in the two-dimensional setting. Similarly, each element of the spatial layout of the system represents a region. As this is a one-dimensional setting the neighbors vector for a region is represented as the regions immediately adjacent to the current region, r_{i-1} and r_{i+1}. Temporally the system is also simulated as conducting over 24 time slots to represent 24 hours.

The user model is equivalent to the previously described setting in terms of the budget, user cost model, providing offers to users, and the utility calculation. Similarly, the service level of the system is the primary metric seeking to be optimized. Users are simulated as tuple of length two with a continuous arrival interest location and destination interest location each prescribed by a related distribution. Bikes are simulated as being placed in regions according to a distribution at the start of the simulation. The supply at each region changes throughout the simulation dynamically according to user behavior.

4.4.2 System Simulation

The main focus of this setting is to compare the effects of varying user interest and supply distributions. These distributions were determined by beta-variate distributions. Beta-variate distributions allow for flexibility in the shape of the distribution as determined by two parameters $\alpha > 0$ and $\beta > 0$. The beta distribution function is shown below in Equation 17 and the beta function utilized in the distribution function is shown in Equation 18.

$$Beta(\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)} \tag{17}$$

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt \tag{18}$$

Specifically in experimentation 5 beta-variate distributions were selected B_1=B(α=0.5,β=0.5), B_2=B(α=5,β=1), B_3=B(α=1,β=3), B_4=B(α=2,β=2), and B_5=B(α=2,β=5). All combinations of these distributions were utilized in experimentation to determine, initial supply distribution, user arrival interests and user destination interest for a total of 125 combinations or scenarios. By varying the distributions that determine user behavior and supply distribution will provide insight on the impact of variations in these crucial distributions.

In the two-dimensional problem setting varying representations for the agent to observe and interact with the environment are evaluated. In this one-dimensional problem setting evaluation is primarily focused on varying distributions in a static environment. Only the supply at each region is provided as a state representation. Further, an incentive

is attached to each region to incentivize unserviceable users to move away from their current region in any direction at every timeslot.

Chapter 5

System Simulation

Bikeshare operators often publish their data publicly according to the General Bikeshare Feed Specification (GBFS) outlined by the North America Bike Share Association (NABSA). This specification is an open data standard for bikeshare systems designed to allow for internal and external parties to better communicate and understand bikeshare system data. The primary information conveyed in a bikeshare's GBFS feed is the station information JSON and station status JSON. The station information JSON includes a record of static information for each publicly available station detailing its location, name, identifier, and total capacity. The station status JSON includes dynamic information for each station regarding the number of currently available bikes and docks. In experimentation, the station information JSON records are used to understand the layout of the system and match stations to regions.

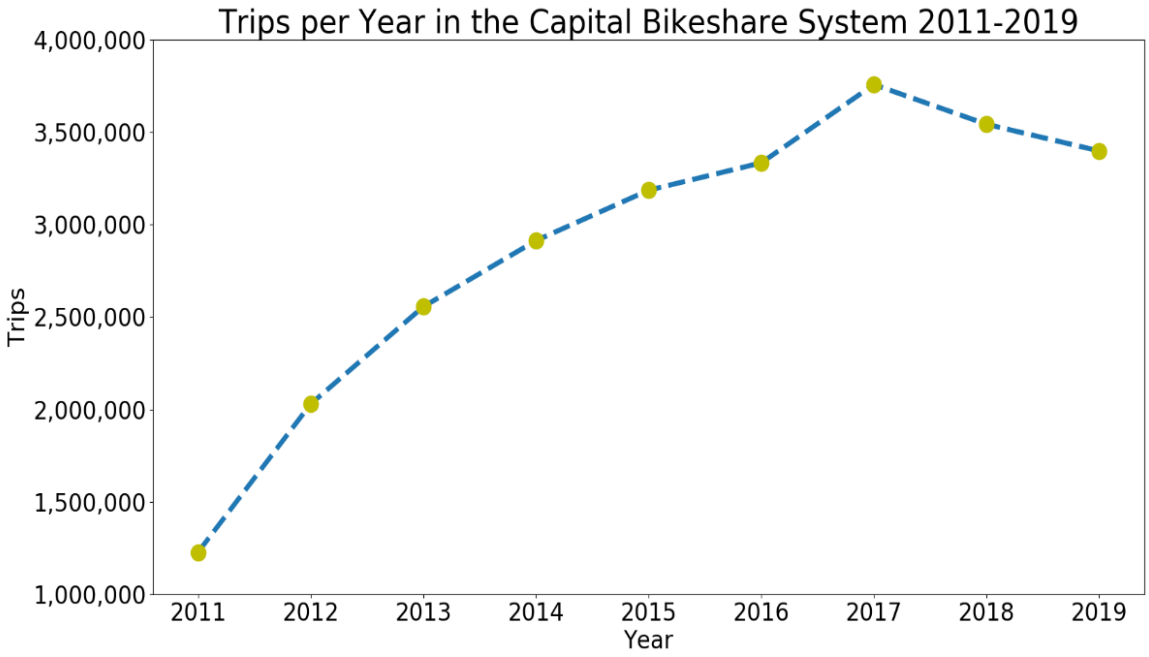
To understand and simulate user behavior in analysis and experimentation bikeshare system trip data has been acquired from Washington D.C.'s Capital Bikeshare website under their System Data – Trip Data repository [32]. Here the Capital Bikeshare system operators maintain an index of files that detail information about each trip that occurred over a specific period of time. These trip data files include the duration, start/end datetime, and start/end latitude/longitude locations for each trip. This data is used to parse various user behavior distributions for generating simulations in experimentation and evaluation.

5.1 User Demand Interests

It is important to understand how the popularity of the Capital Bikeshare system has changed over time and continues to change based on temporal circumstances across various seasons, months, weekdays, and hours. In this section, various temporal slices will be analyzed across collected Capital Bikeshare data.

Figure 2

Trips per Year in the Capital Bikeshare System

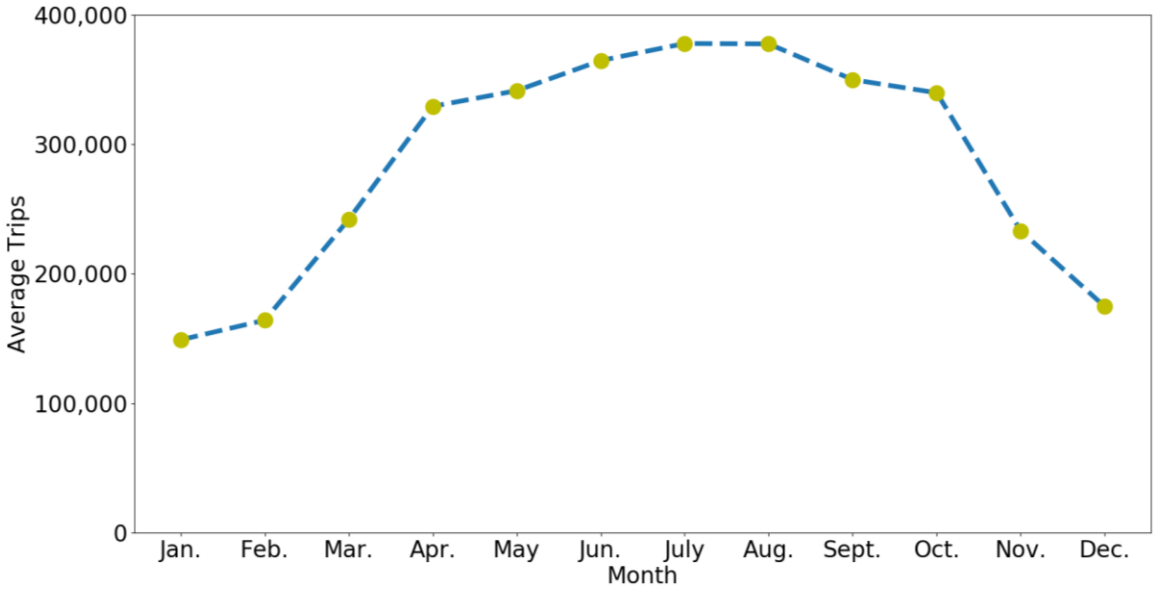


In Figure 2, we see the number of trips per year in the Capital Bikeshare system. The number of trips per year for the Capital Bikeshare system begins to plateau in the year 2015 with the number of trips per year settling to roughly three and a half million

after large year of year growth 2011 through 2015. Further, due to the COVID-19 pandemic and associated countermeasures bikeshare data for 2020 will not be included in analysis and experimentation due to its radical deviation in user behavior distributions relative to prior years. From now on only the Capital Bikeshare trip data inclusively between the years 2015 and 2019 will be used in analysis and experimentation, due to a similar number of users across this time span year over year.

Figure 3

Average Trips per Month in the Capital Bikeshare System 2015-2019

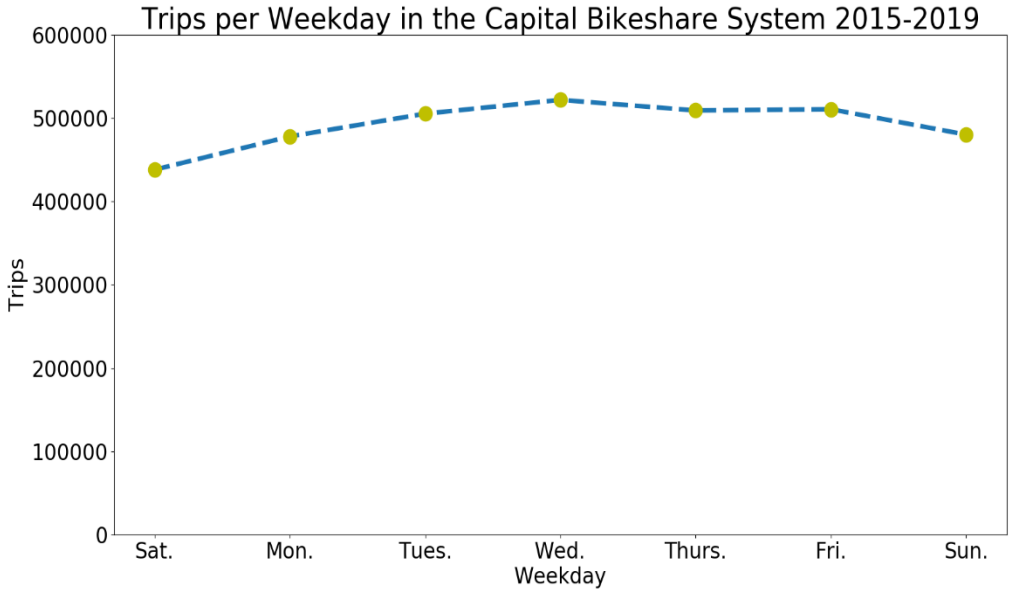


The average number of trips in the Capital Bikeshare system is also highly contingent on seasonal and monthly patterns as seen in Figure 3. The average number of trips per month peaks in the summer months June, July, and August, with almost triple

the number of trips relative to the winter months. The system also maintains relatively high levels of user activity in the late spring and early Fall months of April, May, September, and October. The average number of trips is greatly reduced between the late fall to early spring months November through March, attributable to user preferences regarding inclement weather and reduced tourism in these months. Likewise, from now on only the trip data inclusively between the months April to October will be used in analysis and experimentation, due to high levels of user activity.

Figure 4

Trips per Weekday in the Capital Bikeshare System 2015-2019

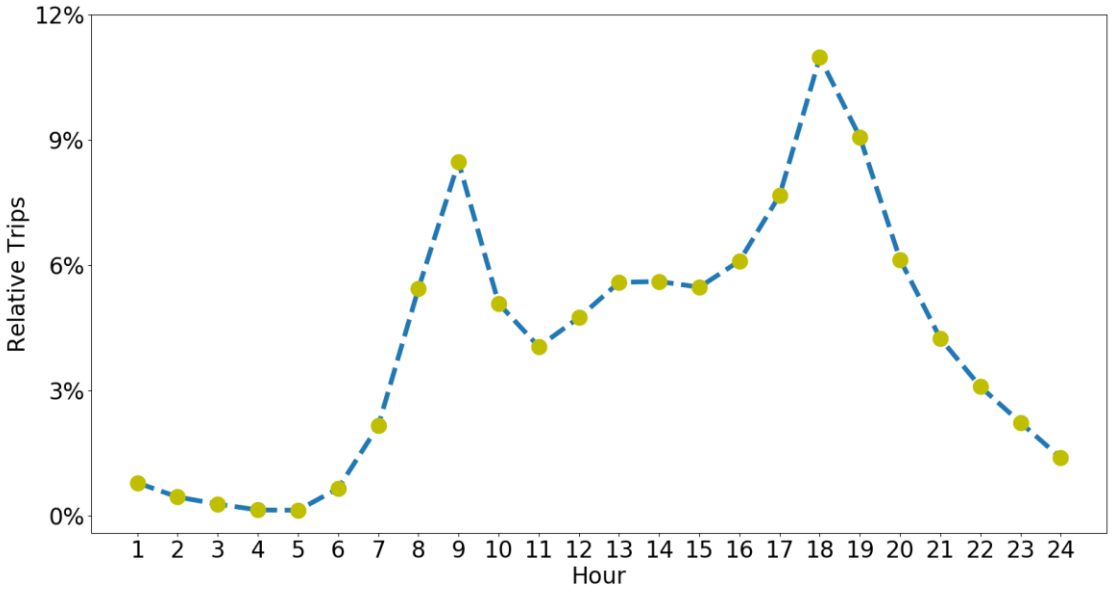


The average number of trips per weekday shows an unexpectedly little amount of variation. It would be expected that there would be fewer trips on weekends due to a reduction in commuting, however as seen in Figure 4 there is little variance in demand

per weekday. Perhaps the reduction in standard commutes is overcome by an increase in weekend leisure activities. Likewise, we will not consider special cases for any day of the week in experimentation.

Figure 5

Average Trips per Hour in the Capital Bikeshare System 2015-2019



The average number of trips per hour as shown in Figure 5 shows temporal behavior that is expected in relation to standard commute patterns. There is a large peak in demand in the mid-morning and mid-afternoon hours of 8am to 10am and 5pm to 7pm as would be expected of users commuting to and from their workplaces and their residences. There is modest mid-day activity between the hours of 11am to 4pm likely due to general tourism and leisure activities. After 7pm, as would be expected through nighttime hours, the system's demand rapidly drops resulting in very little activity

between the hours of 11pm to 6am. This hourly activity distribution will be accounted for in experimentation. The number of users per day will not be distributed uniformly across all time slots, instead they will be distributed according to this hourly activity distribution.

To summarize, based on this temporal activity analysis, data used for further analysis and in experimentation will be between the years 2015 to 2019 and will consist of the warmer months April through October. Further, the distribution of users per hour throughout the day will be accounted for. Additionally, there will be no special consideration applied to the day of the week.

5.2 Distributions

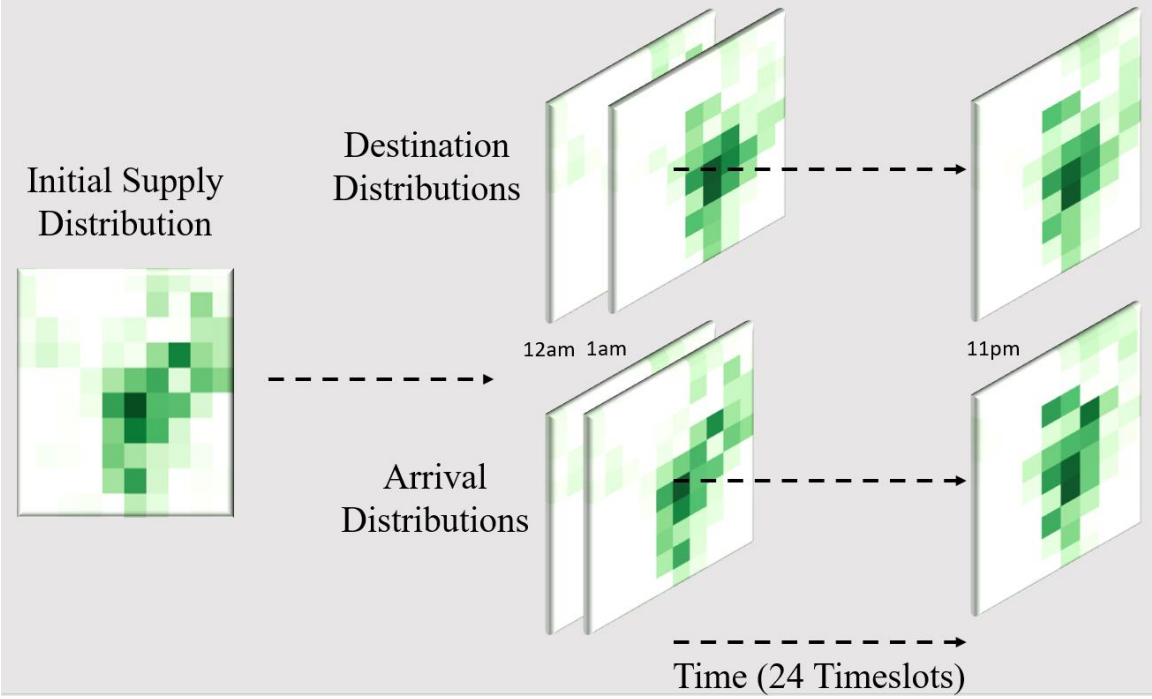
Users are simulated as user objects introduced to the system as represented by a tuple of length two. The first element being their arrival region interest and the second their destination region interest represented by each region's respective integer identifier. For example, a user may be represented as (82, 3) where the user's arrival region is 82 or $r_{8,2}$ and their destination region is 3 or $r_{0,3}$. Both a user's arrival region and destination region interests are determined by distributions based on the hour in which a user appears.

The initial supply distribution is calculated based on placing bicycles at each region throughout the system according to a set supply distribution. This distribution is then used to distribute the number of bikes provided by the supply parameter to the environment i.e., if the system is to have 4,500 bikes then so many bikes will be distributed proportionally throughout the regions of the system before the simulation begins. If the distribution is set to purely uniform, then each of the 100 regions would

receive 45 bikes. The supply distribution is then organically altered as the simulation executes through user arrival and destination interests and behaviors, as additionally influenced by an agent offering incentives.

Figure 6

Illustration of the Daily Initial Supply, Hourly Destination, and Hourly Arrival Distributions for System Simulation



In Figure 6, we illustrate how daily initial supply, hourly destination, and hourly arrival distributions are used in environment simulations. The leftmost distribution is the supply distribution responsible for the initial resource supply layout of the system. The following distributions to the right of Figure 6 show distributions responsible for

determining user arrival and destination interests. There is a distribution for each hour for both arrival and destination interests resulting in 48 total user interest distributions and one supply distribution in the environment generation system.

The initial supply distribution used for Capital Bikeshare simulation is shown as the left most distribution in Figure 6. This distribution is based on the total relative destination popularity of each region in the hour 6 pm, plus a small increase as to distribute supply more evenly. This is described in Equation 19 where *H* is set to 18 (6 pm) and *N* is set to 100 (100 regions). The hour 6 pm was selected as it is the last very popular hour for user activity and is likely to have a large impact on the next day's initial supply. Conversely this implies that the previous day's 6 pm destination activity has a large impact on the initial supply distribution when a simulation begins. Then one is added to each element, to give each region some probability of receiving a bike. This extra step of giving each region some supply allows for more opportunities to reroute users. Lastly, this matrix is normalized to create this supply distribution.

$$\frac{1 + Number\ of\ Destinations\ for\ r_{ij} in\ H}{N + \ Total\ Number\ of\ Destinations\ in\ H} \ \forall\ regions\ r_{ij} \tag{19}$$

A user's arrival interest and destination interest are both calculated based on the hour in which the user appears. Each hour indexes two distributions each with the probability of a user either arriving at or selecting as a destination for each region. The user arrival distribution is calculated based on historical user arrival patterns as shown in Equation 20. The user destination interest distribution is calculated similarly, where instead of utilizing historical arrivals, the distribution is calculated based on historical

destinations. P(A | H) shows the conditional probability for determining user arrival location interest, where A is the arrival region and H is the hour in which the user appears

$$\frac{\sum arrivals\ for\ r_{ij} in\ H}{\sum total\ arrivals\ in\ H}\ \forall\ regions\ r_{ij} \forall\ hours\ H \tag{20}$$

5.3 Limitations

It is important to recognize that while this simulation environment is consistent with much of the related literature, as well as introducing subtle improvements, it still has many limitations and possible improvements. This simulation assumes that when a user occupies a resource that the resource will not appear in its destination region until the start of the next hour time slot. This is not an accurate representation of the trip duration dynamics of a bikeshare system as bicycles are constantly arriving and departing due to trips often being less than a half-hour in duration. This limitation is introduced as it greatly reduces the computational complexity of simulation and is assumed to not greatly impact simulation effectiveness.

In experimentation, the Capital Bikeshare system is divided into a 10x10 grid resulting in 100 equally sized regions. The largest trade-off in determining the input size of the system's grid representation is the ratio of regions whose neighbors vector length is less than 4. Under the 10x10 grid size, there are 32 edge regions and 4 corners resulting in 36% of the regions having a neighbors vector less than 4. For comparison, under a 7x7 grid size there are 26 edge and corner regions resulting in 53% of regions with a neighbors vector less than 4. With a grid of 100 regions, there is also a reasonable amount of separation between where users are typically interested in departing versus where they are intended to commute to. This is an important distinction as it is redundant to offer a

user an incentive to walk to a region that they were otherwise intending to travel by bicycle to.

The entire Capital Bikeshare system spans a wide area from Dulles International Airport, to Alexandria, to Capital Heights, to Gaithersburg. The size of the Capital Bikeshare system is reduced to its central area from Arlington to slightly South of Mt. Rainer. Specifically, the system is reduced from spanning from latitudes and longitudes (38.783, -77.368) to (39.124, -76.826) to spanning the latitudes and longitudes (38.875, -77.1) to (38.935, -76.98). This reduced system size still captures roughly 90% of trips in user activity data and is less than half of the original spatial area covered by the system. This reduction in coverage area makes regions more meaningful as they better capture specific locations, and it is more likely that a user will travel to a region further than a neighboring region.

Another limitation of this simulation is its assumption that all users will behave according to the same cost model. In practice, the walking distance cost for each individual is likely to have a wide variance dependent on a variety of internal and external factors to the user. Someone engaging in exercise will likely need a much lower incentive than someone who is commuting to their workplace in order to depart from a bicycle further away. Likewise, due to various weather circumstances or road blockages a user may find it more difficult than our model would anticipate to move to a new location. This is not a significant limitation as it is likely that the cost model will capture an approximation of average user behavior and given such a large amount of simulated users will be representative.

Chapter 6

Representations Description

In this chapter, descriptions are provided of the various representations that will be utilized in experimentation. Five representations are employed for the environment's state and action mechanism to observe performance impacts, as summarized in Table 1 and Table 2.

Agents received the reward signal reduced unservice level, shown in Equation 21. This is the percentage reduction in unservice level, directly derived from the service level metric, there will not be a case where a reduction or increase in one is not correlated with a reduction or increase in another. In reinforcement learning, it is critical that the reward function reflects the overall goal intended to be achieved within an environment. For example, for an agent to learn the game of chess the reward function should reflect whether the agent achieves victory or is defeated and not other metrics such as the number of pieces captured. While the system's service level metric is the most critical measurement, this metric will typically be within a very small range whereas the reduced unservice level will typically be within a much larger range. If service level alone were relied upon as a reward function its small range would be difficult for an agent to interpret, therefore the reduced unservice level metric is utilized.

$$\text{Reduced Unservice Level} = \text{RUL} = 1 - \frac{1 - SL_{with\ agent}}{1 - SL_{no\ agent}} \tag{21}$$

Table 1

State and Action Representation Symbols

Symbol	Description
S_t	Supply at each region at time slot t
A_t	User arrivals at each region during time slot t
D_t	Trip destinations for each region during time slot t
U_t	Unservice level for each region during time slot t
E_t	Expense at each region during time slot t
A’	Known trip arrivals for each region during time slot t
D’	Predicted trip departures for each region during time slot t
o^t_{ij}-	Offer to move to any neighboring region from region r_{ij} during time slot t
o^t_{pq+}	Offer to move to region r_{pq} during time slot t
o^t_{pq}+’	Offer to change destination to region r_{pq} during time slot t

Table 2

Representation Summaries

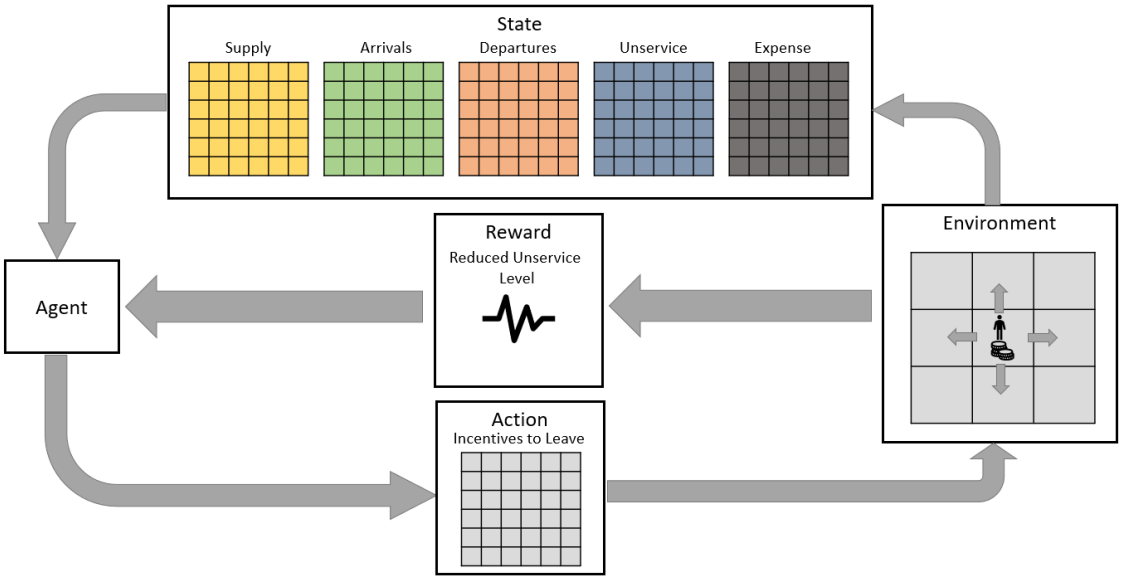
Representations	State Representations	Action Mechanisms
SADUE-A	$(S_t, A_{(t-1)}, D_{(t-1)}, U_{(t-1)}, E_{(t-1)})$	$(o^t_{11}\text{-}, o^t_{12}\text{-}, \dots, o^t_{nm}\text{-})$
S-A	(S_t)	$(o^t_{11}\text{-}, o^t_{12}\text{-}, \dots, o^t_{nm}\text{-})$
S-A+	(S_t)	$(o^t_{11}+, o^t_{12}+, \dots, o^t_{nm}+)$
SA’D’-A+	(S_t, A_t', D_t')	$(o^t_{11}+, o^t_{12}+, \dots, o^t_{nm}+)$
SA’D’-A+D+	(S_t, A_t', D_t')	$(o^t_{11}+, o^t_{11}+', o^t_{12}+, o^t_{12}+', \dots, o^t_{nm}+, o^t_{nm}+')$

6.1 Representations Without Using Predictive Model

6.1.1 SADUE-A

Figure 7

SADUE-A Representation Diagram



In Figure 7, a diagram of the first representation named SADUE-A is shown, this representation is largely inspired by the framework proposed by Pan et al. [6]. Under the SADUE-A representation a high-dimensional state space is utilized which captures a relatively large amount of information about the observed mobility-system environment. The first part of the representation's name, SADUE, stands for its state representation a tuple (S_t, $A_{(t-1)}$, $D_{(t-1)}$, $U_{(t-1)}$, $E_{(t-1)}$). S_t is a matrix which represents the current supply at each region, it is reasonably assumed that this is the most important feature to represent

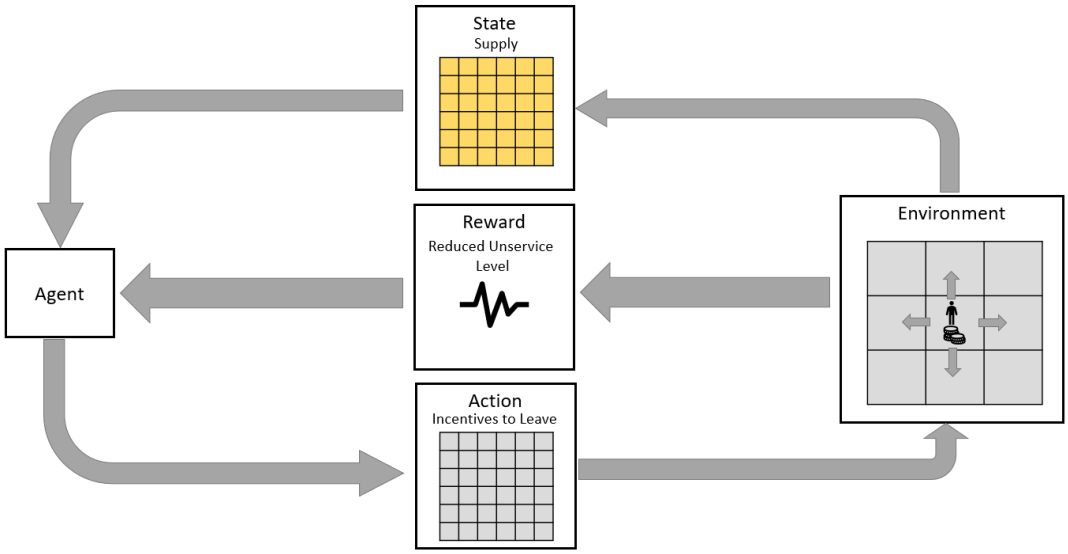
the environment. The following matrices $A_{(t-1)}$ and $D_{(t-1)}$ respectively represent the number of resource arrivals and departures during the previous time slot (t-1), this is intended to inform the agent what may happen in the following time slot. The matrix $U_{(t-1)}$ is a matrix which represents the unservice level at each region during the previous time slot (t-1), this is intended to highlight problematic regions. The last matrix $E_{(t-1)}$ represents the expense at each region during the last timestep, further highlighting problematic and high demand regions. This is a very high dimensional state space with high variability, that will require extra training time for an agent to understand key features of the state space.

The action representation of SADUE-A is denoted as simply A, this action mechanism is designed based on the mechanism proposed by Pan et al. [6]. Under the action mechanism A, the agent applies an offer at the start of every time slot to incentivize users to move away from their current region to any neighbor. This action is represented as the vector (o^t_{11}-, o^t_{12}-,… , o^t_{nm}-), the - signifies that the agent is incentivized to move away from their current region in any direction for the same offer. The user in region r_{ij} will consider the offer o^t_{ij} as it relates to movement cost to each of its neighbors $N(r_{ij})$, the offer vector the user receives is of the form (o^t_{ij}, o^t_{ij}, o^t_{ij}, o^t_{ij}).

6.1.2 S-A

Figure 8

S-A Representation Diagram



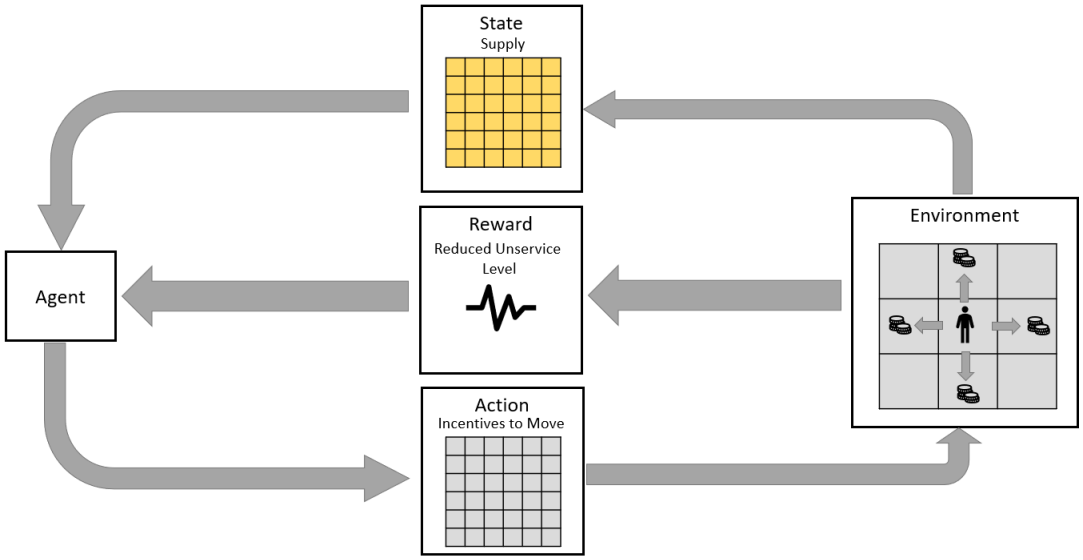
In Figure 8 a diagram of the next representation S-A is shown, this representation is designed to observe the effects of a reduction in SADUE-A's state space. High dimensional state spaces typically cause a longer convergence time in machine learning based algorithms. As the input space grows there are more variables and combinations thereof to fit to the desired output. Worse in our problem setting it is reasonable to assume that each feature is extremely non-uniform in importance, thus requiring extra training time as the initialized model will give each feature roughly equal weighting. Under S-A the state space is reduced to only the current regional supply matrix S_t, omitting information ('ADUE') that only provides theoretically marginal information

about the environment. Information such as the previous time slot's user arrivals and departures does provide some information regarding an expectation on activity in the current time slot, this will be missed though there will be a particular focus on the current supply. However, the unservice level and expense at each region is likely to only challenge the agent without providing significant usable information. The action mechanism A is the same as the action mechanism used under SADUE-A, for consistency in evaluating only the reduced state space.

6.1.3 S-A+

Figure 9

S-A+ Representation Diagram



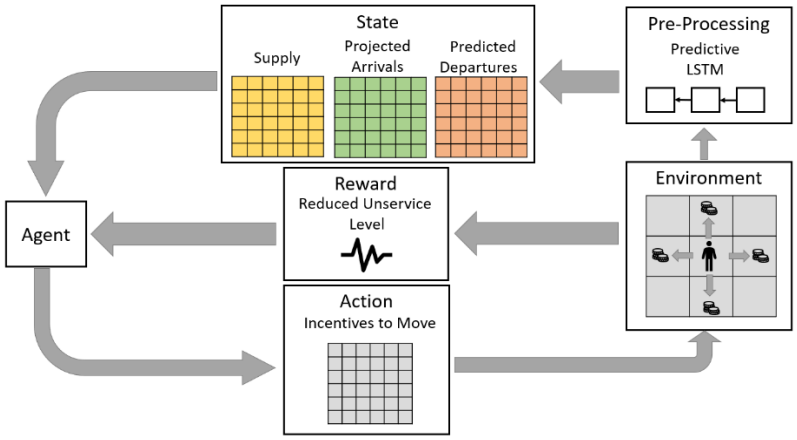
In Figure 9 a diagram of the S-A+ representation is shown, this representation seeks to investigate the effectiveness of a theoretically more advantageous action mechanism A+, while maintaining the state space S found to be effective in comparison between SADUE-A and S-A. Under this new action mechanism, A+, the agent applies a static incentive to incentivize users to move to each specific neighboring region rather than to only move away from their current region. This action mechanism gives the agent more control over which regions the users move to or avoid. This action representation is represented as the vector ($o^t_{11}+$, $o^t_{12}+$, …, $o^t_{nm}+$), the + signifies that the agent is incentivized to move to a region rather than away from a region in any direction. A user in region r_{ij} will consider the offer to move to each neighboring region in $N(r_{ij})$ as it relates to movement cost to each, the offer vector is of the form ($o^t_{i+1,j}$,$o^t_{i-1,j}$,$o^t_{i,j+1}$,$o^t_{i,j-1}$).

6.2 Representations Including a Predictive Model

6.2.1 SA'D'-A+

Figure 10

SA'D'-A+ Representation Diagram



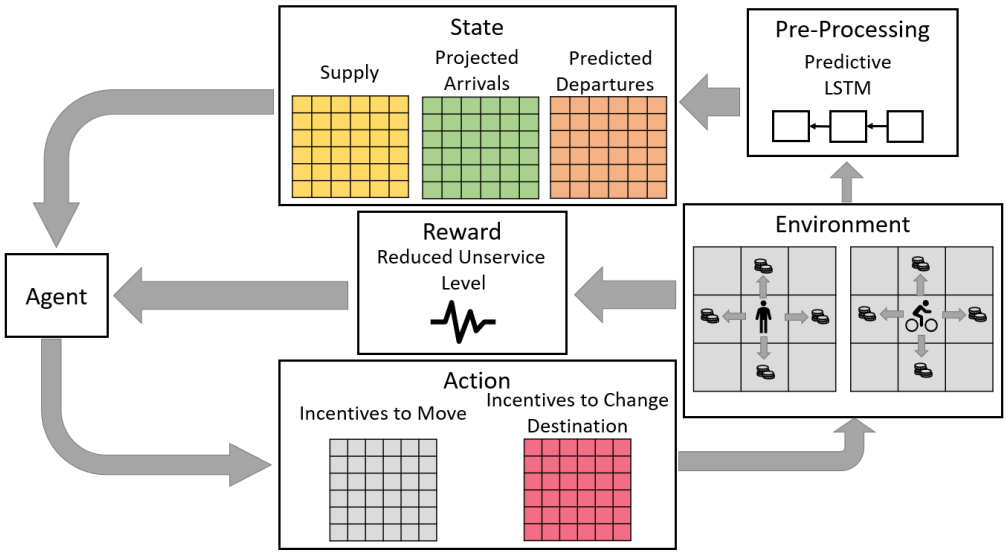
In Figure 10 a diagram of the SA'D'-A+ representation is shown, this representation incorporates user behavior information to supplement the state representation of the environment. The term A' represents the matrix A_t which quantifies the resource arrivals at each region for the current time slot t. A_t is known as users signify their destination regions, i.e. after the activity of users in time slot t-1 A_t is known. This does not provide significant novel information as these users already arrive and the supply is resolved for each respective arrival and region prior to the agent allocating a new incentive scheme. The term D' represents the matrix D_t' which quantifies the predicted user departures from each region in the current time slot t. This is intended to

aid the agent in understanding which regions will have a high demand as to more effectively allocate a budget.

6.2.2 SA'D'-A+D+

Figure 11

SA'D'-A+D+ Representation Diagram



Lastly in Figure 11 a diagram of the representation SA'D'-A+D+ is shown, this representation seeks to investigate the benefits of a further expanded action mechanism A+D+. Under the A+D+ action mechanism the agent will apply not only a static incentive to move toward each respective neighboring region on user arrival, but as well static incentives for a user to change their destination region. This action mechanism provides the agent with even greater control over user behavior, however it also presents a greater challenge of budget management. The action representation as applied to each

region is as follows, (o^t_{11}+, o^t_{12}+', o^t_{12}+, o^t_{12}+', …, o^t_{nm}+, o^t_{nm}+'). As in the action mechanism A+ users receive an offer vector to modify their arrival region of the form ($o^t_{i+1,j}$, $o^t_{i-1,j}$, $o^t_{i,j+1}$, $o^t_{i,j-1}$) to each region in the set N(r_{ij}). Extending upon A+ the signifier D+ indicates users also receive an offer vector to similarly modify their destination region to a neighbor of their destination region r_{pq} in the set N(r_{pq}) of the form ($o^t_{i+1,j}$', $o^t_{i-1,j}$', $o^t_{i,j+1}$', $o^t_{i,j-1}$'). This representation utilizes the same state representation of SA'D'-A+ in SA'D', both utilize the same pre-trained Long-Short Term Memory network (LSTM) D_t' predictive model (See Section 6.2.3) [33]. This state representation was reused due to its theoretical benefit of adding meaningful information to the environment's state space.

6.2.3 Predicting Departures Using Long-Short Term Memory Network

The SA'D' state representation utilized in the SA'D'-A+ and SA'D'-A+D+ representations utilize a LSTM as a predictive model for approximating D_t'. D_t' is a matrix of the predicted frequency of departures per region in time slot t. In order to account for a varying number of users the model itself outputs a normalized D_t' that is then multiplied by the number of users for that time slot. This approach allows the same model to be used to approximate D_t' for a varying number of user level parameters *N*, rather than training a new model for every user level parameter setting. The total number of users for the next time slot is assumed to be known.

The LSTM model is comprised of ten LSTM units that feed into two dense layers that are one-hundred nodes each. The input to this model is the previous three time slots' frequency of departures per region (D_{t-1}', D_{t-2}', D_{t-3}') normalized and the output is the predicted frequency pf departures per region normalized for the current time slot D_t'.

This output D_t' is then multiplied by the number of users for the predicted time slot N_t and is then presented to the agent as state information.

The LSTM model was trained by generating 100 days of user arrival interest matrices, this equates to 2400 D_t' matrices. Note that supply is assumed to be infinite such that user arrival interests per region equate to user departures per region. These matrices are ordered as sets X and Y where beginning with D_3' every D_t' is given as a target matrix Y with the given X features (D_{t-1}', D_{t-2}', D_{t-3}'). A randomized 80-20 train-test split is utilized to then train and evaluate the LSTM. It is the case that based on mean squared error it would seem that the LSTM performs poorly in estimating D_t', however on observation D_t' matrices produced do adequately identify areas of high arrival interest and do improve performance as seen in SA'D'-A+ versus S-A+. Performance of SA'D'-A+ when utilizing the LSTM predictive is only slightly lagging the performance when true D_t' values were used. Likewise, in experimentation the SA'D' state space will utilize the LSTM predictive model.

Chapter 7

Experiments and Results

This chapter summarizes a set of experiments and their results to understand the performance of the various reinforcement learning representations for the bikeshare rebalancing problem, as previously outlined, on various performance metrics and constraints.

7.1. Metrics and Baselines Description

The primary metric utilized to measure performance is service level. This is the ratio of users who have requested a bicycle that were able to receive one. This is the most critical metric for capturing user experience in the real system as this most directly correlates with service usability.

$$\text{Service Level} = \text{SL} = \frac{fulfilled\ user\ requests}{total\ user\ requests} \tag{22}$$

Our secondary metric used in evaluation is improvement in service level. This is the percentage improvement in service level that the agent provides versus if there was not an agent. Though being derived from service level does not provide significant additional information.

$$\text{Improved Service Level} = \text{ISL} = \frac{SL_{with\ agent} - SL_{no\ agent}}{SL_{no\ agent}} \tag{23}$$

In experimentation, two baselines are utilized to provide bounds on the expected performance of varying representations. The no agent baseline is utilized to provide a lower bound on representation performance. This baseline represents system performance

if there were no agent providing incentives. It is possible for agents to underperform the no agent metric in the case that changing the behavior of one or more users then causes a greater number of users to lose service at a later time step. In practice an agent should always show at least a slight improvement over the no agent baseline.

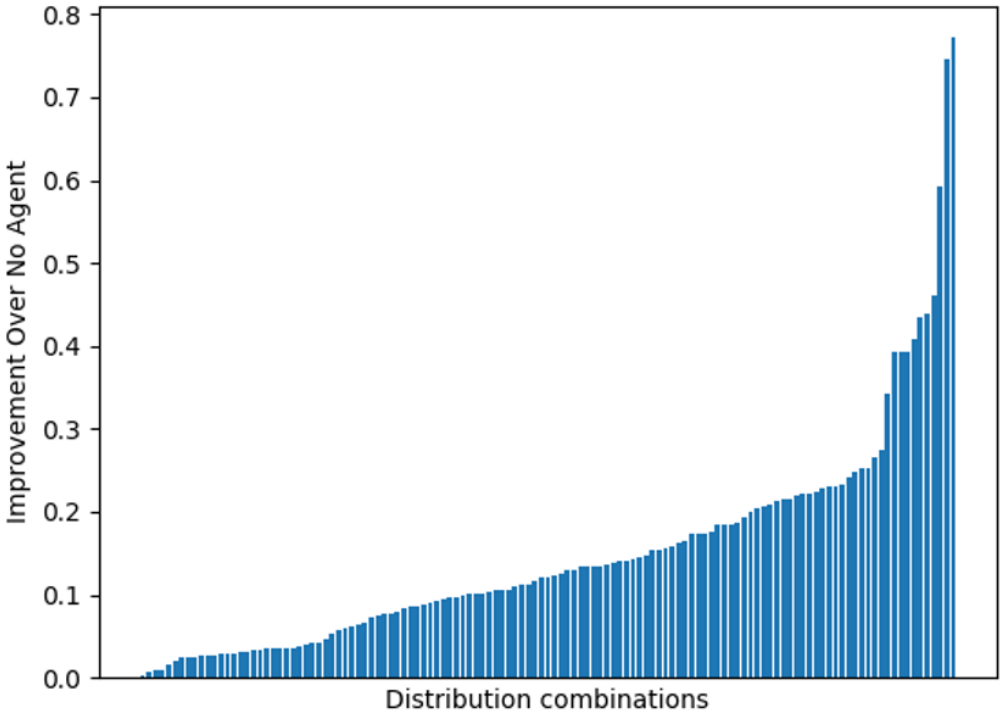
The empirical optimal baseline is utilized to provide an upper bound on representation performance. This baseline is created by always and exclusively providing users maximum offers to move in any direction without a budget constraint. This provides an empirical upper bound on agent performance considering that some users will naturally not be serviceable due to supply constraints. It is possible, though unlikely, for agents to outperform this baseline. For example, particular changes to the supply distribution caused by the agent's alterations may result in naturally unserviceable users at later timesteps being able to receive service.

7.2 One-Dimensional Scenario Results

In this subsection, we present preliminary results on the defined one-dimensional scenario (see Section 4.4) to demonstrate the feasibility and usefulness of reinforcement learning-based incentive scheme for the bikeshare rebalancing problem. In experimentation, the effects of a varying budget and the interactions of varying distributions were observed. Three reinforcement learning algorithms (PPO, ACKTR, and TRPO) were utilized in this early setting due to its speed and simplicity. The system size parameter was set to 10 kilometers which in turn created an environment with 10 regions. The supply of bicycles was set at 100 and the number of users per hour was static at 30. Three budgets (100, 200, 300) were also utilized in experimentation.

Figure 12

Performance Across Distributions



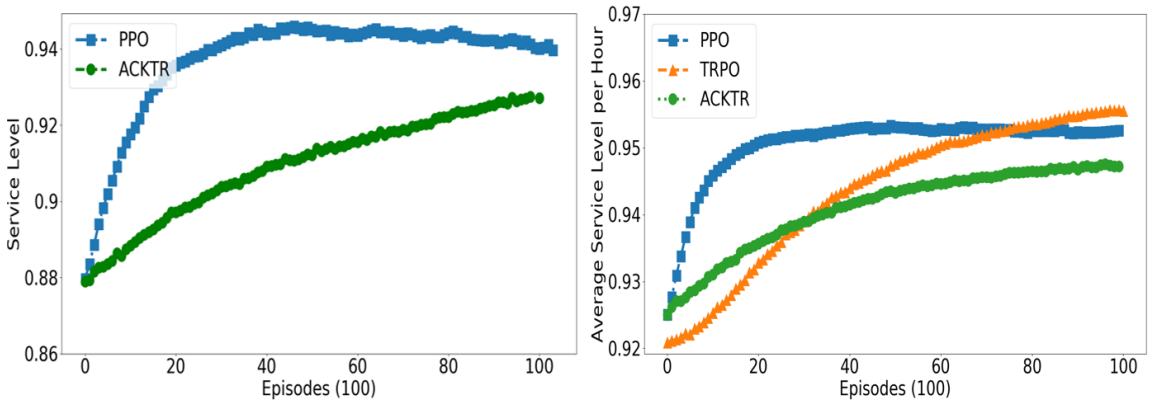
In Figure 12 the performance of each distribution combination is shown as measured by the service level improvement across all agents and budgets. The four distribution combinations with the worst service-level improvement are in order B_2-B_2-B_2, B_5-B_5-B_5, B_4-B_4-B_4, B_3-B_3-B_3. This is interpreted as when all three distributions are equivalent the system naturally maintains balance, and there is little ability for an agent to improve the service level. The fifth distribution overlap, B_1-B_1-B_1, has the eighth lowest service-level improvement. The distribution combinations, B_5-B_2-B_5(0.46), B_4-B_2-B_5(0.59), B_2-B_5-B_2(0.75), B_1-B_5-B_2(0.77), have the highest service-level improvement. This matches expectations that an unbalanced arrival and destination distribution set leads to a large service-level improvement. However, this does suggest that a match

between the supply and destination interest distribution does not necessarily lead to a more naturally balanced system as a match between supply and arrival interest would.

7.3 Reinforcement Learning Algorithm Comparison

Figure 13

Comparison of Reinforcement Learning Algorithm Performance



In Figure 13, a performance comparison is shown of the three reinforcement learning algorithms: Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), and Actor-Critic with Kron-factor Trust Regions (ACKTR). On the left, a comparison is shown between PPO and ACKTR within the described problem setting (see Chapter 4. Problem Setting). On the right, a comparison is shown between PPO, TRPO, and ACKTR, in a problem setting that is similar to the one described however each game utilizes a static set of users and the A'D' state representation component does not predict D_t', but rather uses the known user demand per each region. Both experiments are shown as the implementation of TRPO utilized in the latest experiments was unable to process LSTM predicted outputs for representations that

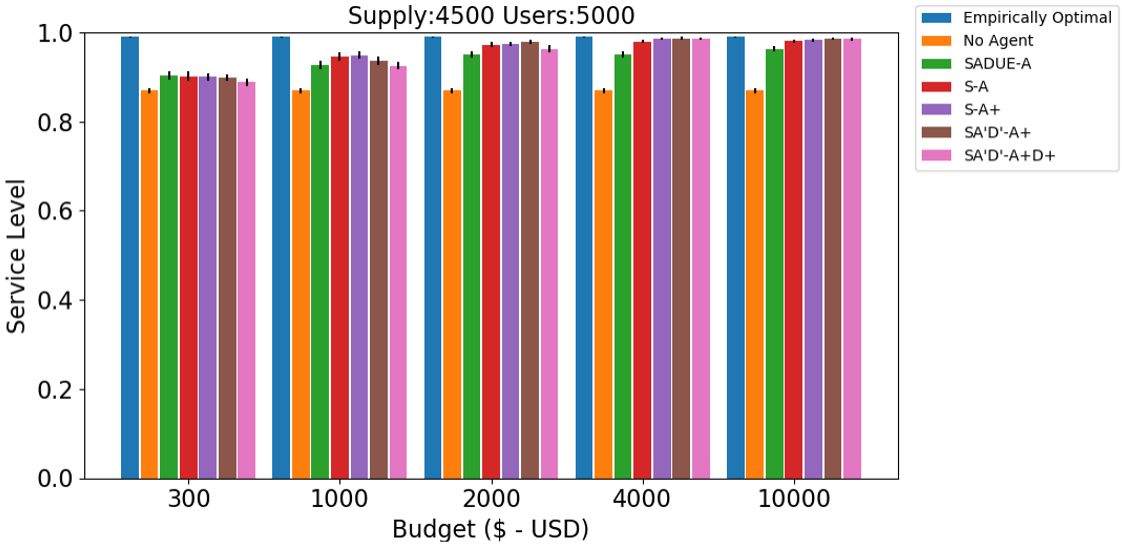
utilized the A'D' as part of their state. Given the similarity in performance between PPO and ACKTR across both experiments it is a reasonable assumption that TRPO would perform similarly.

These algorithms were tested in a scenario given a supply of 4,500 bicycles, 5,000 users, and a budget of 4,000. This experiment is conducted to determine which algorithm to use in further in-depth experimentation. It is clear to see that PPO has a far superior sample efficiency relative to its counterparts as it reaches convergence after roughly 2,000 timesteps while TRPO and ACKTR both require nearly all 10,000 episodes. This is expected as PPO is an improvement of TRPO itself. All algorithms have a similar ending performance, notably TRPO slightly outperforms PPO, this is most likely due to each of these algorithms having a similar basis in trust-regions and a policy-gradient learning approach. PPO will be utilized in further experimentation due to its impressive sample efficiency, that will result in more reliable results in scenarios that are more difficult and likewise will take more samples to learn.

7.4 Representation Comparison

Figure 14

Performance on a Simple Scenario Across Representations and Budgets

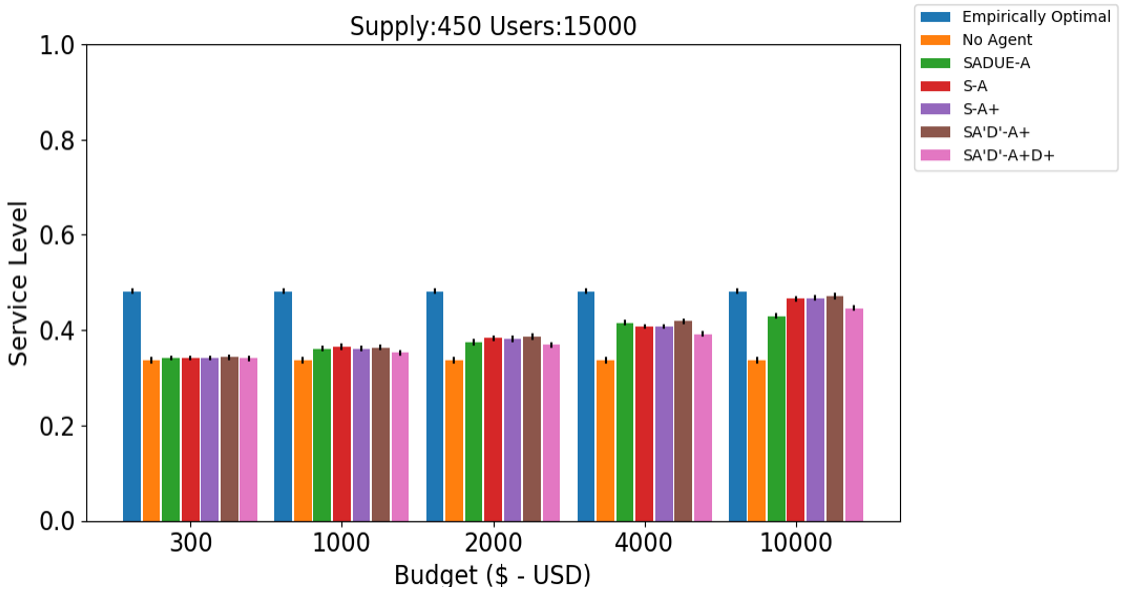


In Figure 14, a performance summary is provided of an experiment in a scenario with low difficulty where the baseline service level is already high at 87%. This experiment shows the performance of various representations across a set of budgets to understand the improvement that each is able to achieve in a baseline high-performing scenario. Under a very low budget, 300, all agents show a similarly small improvement over the baseline service level, on average improving service level to 90%. Under a moderate budget of 1,000 an anticipated trend is achieved where S-A and S-A+ outperform their predecessor SADUE-A. Unexpectedly SA'D'-A+ does not outperform its predecessors until under a high-moderate budget of 2,000 this is likely due to its larger state space being more challenging to learn. This is reinforced by the most complex representation SA'D'-A+D+ requiring a high budget 4,000 and 10,000 to perform in-line

with its predecessors. Overall, in this experiment we see that agents under any representation are able to show a meaningful improvement in service level, even when the baseline system is already able to maintain a high service level.

Figure 15

Performance on a Difficult Scenario Across Representations and Budgets

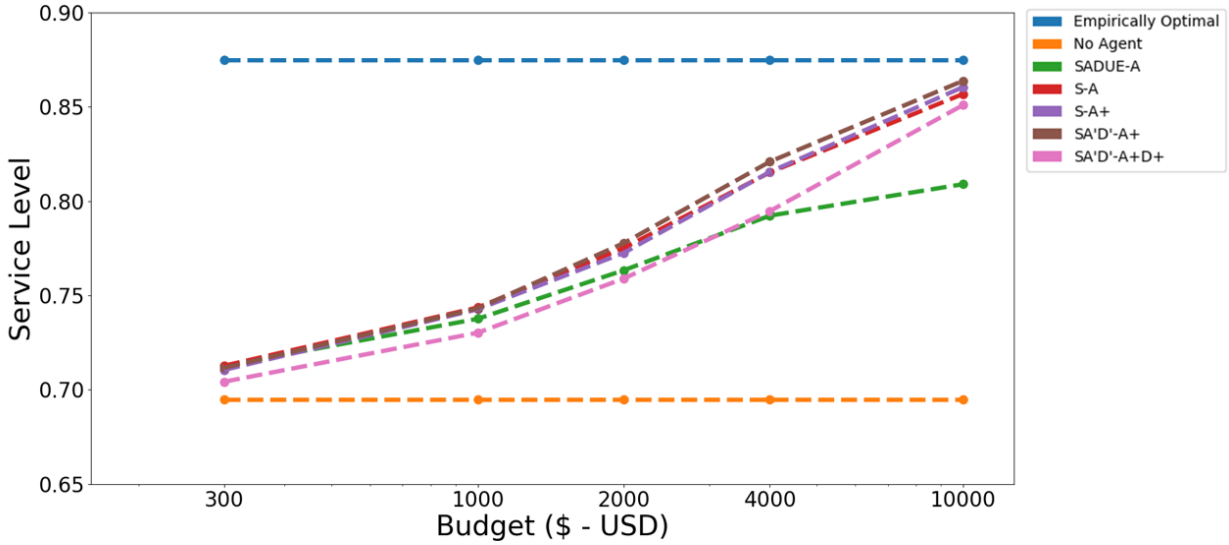


In Figure 15, a performance summary is provided of an experiment where representations were evaluated within a scenario that supports a low baseline service level, at only 33.7%. In this scenario there is little improvement over having no agent when the budget is very low at 300, on average all representations show an improved service level of 0.9%. In this experiment there is strange behavior where representations S-A and S-A+ which typically quickly outperform SADUE-A as budget increases do not outperform noticeably until presented with a very high budget of 10000. This is a

surprising result unseen in other experiments. The SA'D'-A+D+ representation underperforms its predecessors even when given a very high budget of 10,000. Overall, this scenario's difficulty can be readily seen in the performance across representations being very low and inconsistent until given a very high budget.

Figure 16

Performance on All Tested Scenarios Across Representations and Budgets



In Figure 16, a summary of the performance of each representation per each budget across all user demand and supply level parameters. Similar to the experiments summarized above in Figure 14 and Figure 15, we see that on average under a small budget there is an expectedly small improvement in performance at a 2.5% improved service level. Then, as the budget increases significant performance gains are accumulated within each representation. With a very high budget of 10,000 the top four

representations achieve a service level of 86% slightly below the empirical optimal baseline which yields a service level of 87.5% and well above the no agent baseline of 69.5%, the top four approaches show an average improved service level of 23.7% in this scenario.

Notably, the most complex representation SA'D'-A+D+ slightly underperforms the other representations not only under a low budget, but throughout experimentation until supplied a very high budget of 10,000. The lagging performance in SA'D'-A+D+ is likely due to its more complex action mechanism that is both more difficult for the agent to learn to employ, but also more demanding on the agent's budget. Given an agent that is able to optimally utilize each representation, it is most likely that the SA'D'-A+D+ would show the strongest performance due to its added state information in predicted A'D' and its action mechanism provides more opportunity for influencing users.

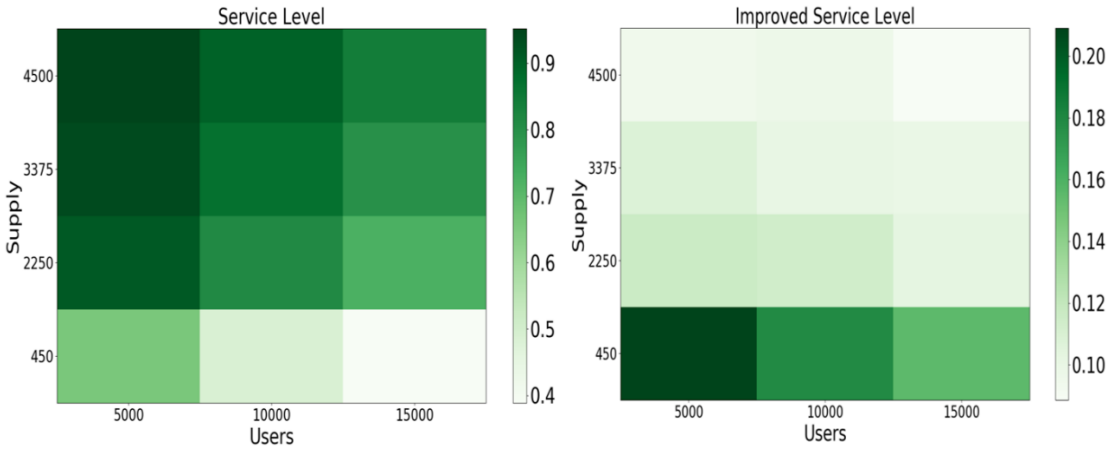
Further, The SADUE-A representation is quickly out-competed by other representations falling slightly behind under the moderate and high budgets of 2,000 and 4,000 before performing much worse given a 10,000 budget. The lagging performance in SADUE-A is likely due to its large state space that is difficult for the agent to effectively learn the key features of the environment. Prior state information regarding arrivals and departures theoretically add little new information to the agent as the current supply is already represented and prior unservice and expense levels per region provide virtually no usable information to the agent and serve as noise. This would explain the success of the reduced state spaces in the other four representations.

Overall, this experiment shows that as budget is increased each representation shows substantial improvement, where the representation SA'D'-A+ shows the highest performance in close comparison with S-A and S-A+. The large state spaces of SADUE-A and the complex action mechanism of SA'D'-A+D+ makes it difficult for the agent to adequately learn the environment, as it must first learn to attribute the importance of the given state or action features. Whereas under the lower and more meaningful state representation and simplified action mechanisms of S-A, S-A+, and SA'D'-A+, the agent performs strongly.

7.5 Constraint Comparison

Figure 17

Performance Comparison Across Users and Supply Parameters



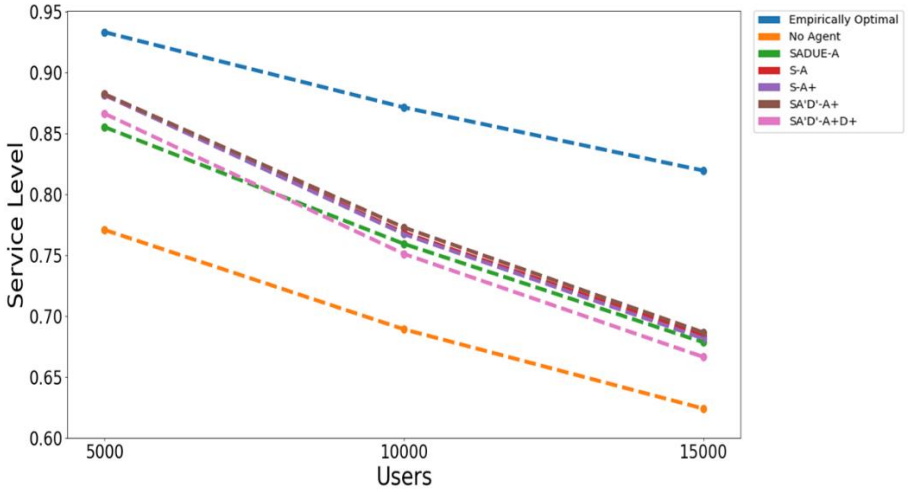
In Figure 17 the average performance over all representations and budgets is compared across combinations of user and supply parameters as measured by service level and improved service level. On the left the performance in terms of service level is

shown, this metric may be misleading as some scenarios naturally produce a high service level. Likewise, on the right the performance in terms of improved service level is shown, this metric is an indicator of how effective an agent is at improving the system versus having no agent.

From the service level comparison, a clear pattern is shown whereas the number of users is increased, and the supply is reduced the service level decreases. This is an expected pattern in line with assumptions regarding the impacts of the supply and user demand parameters. In the improved service level comparison, a strong improvement in performance is shown given a low supply and a low number of users. Further, the improvement in service level is lowest given a high supply and a high user demand. This suggests that an agent is best suited for improving environments with a low supply and struggles most with managing an increasing number of users.

Figure 18

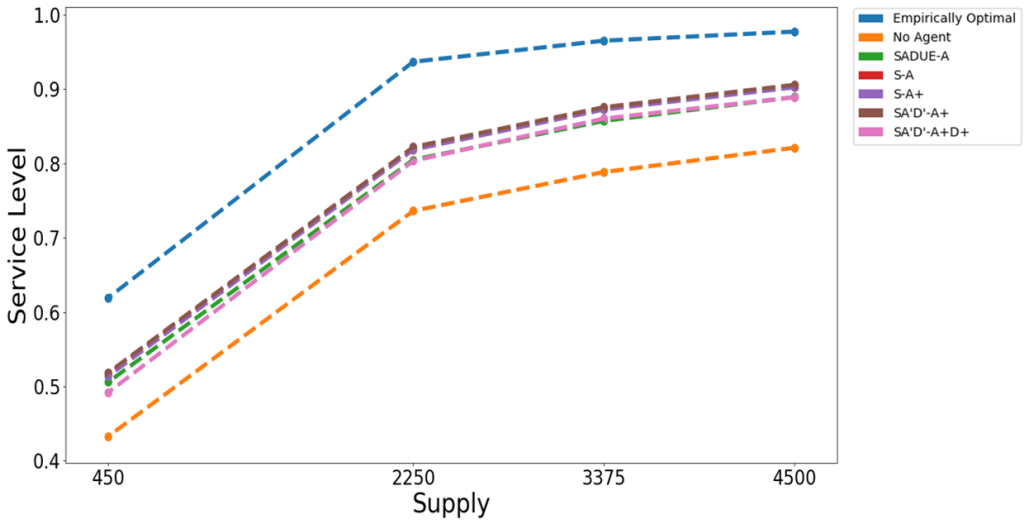
Performance on Varying User Demand Scenarios Across Representations



In Figure 18 the performance impacts of varying the environment parameter number of users on the performance of each representation is shown. Additional environmental constraints evaluated against are all supply levels (450, 2250, 3375, 4500) and budgets 1000, 2000, and 4000. The budgets 300 and 10000 were excluded as these are extreme cases that dilute the result of the more competitive moderate budgets. Under the least challenging user parameter setting of 5000 the top three representations S-A, S-A+, and SA'D'-A+ perform near identically. Then, as the user parameter is increased their performance begins to vary more widely with SA'D'-A+ performing the best under the most challenging number of users setting 15000. The SA'D'-A+D+ representation under the 5000 number of users parameter setting underperforms the tied top three representations, however it does outperform the SADUE-A representation. As the number of users is increased the SADUE-A framework does not decrease in performance as quickly as the other representations, suggesting that its larger state space makes it less sensitive to the number of users parameter.

Figure 19

Performance on Varying Supply Scenarios Across Representations



In Figure 19 the performance impacts of varying the supply level environmental parameter on the performance of each representation is shown. Additional environmental constraints evaluated against are all user levels (5000, 10000, 15000) and budgets 1000, 2000, and 4000 similarly to the varying number of users parameter experiment shown previously. In this experiment, the top three representations S-A, S-A+, and SA'D'-A+ show similar performance improvements as the supply level is increased. The representation SA'D'-A+D+ noticeably underperforms given the very low supply level of 450. As under the supply level of 450 a much larger budget utilization will be needed as more users will need to be incentivized. The SA'D'-A+D+ is more sensitive to budget pressures compared to other representations due to its D+ action mechanism component allowing it to more quickly deplete its budget.

Chapter 8

Conclusion

In this closing chapter I will summarize the meaning behind experiment results and possible avenues for future research to expand upon this work**.**

8.1 Summary

Shared mobility systems, bike-sharing, scooter sharing, etc., are rapidly growing in popularity in densely populated areas, such as cities and university campuses. These systems provide an alternative means of travel that reduces traffic congestion, is environmentally friendly, and promotes exercise. However, these systems often suffer from their resources becoming imbalanced and users being unable to receive service. In shared mobility systems research, there is increasing interest in the use of reinforcement learning techniques for improving the resource supply balance and service level of these systems. These techniques seek to effectively produce a user incentivization policy scheme to encourage users of a shared mobility system (e.g. bikeshare systems) to slightly alter their travel behavior in exchange for a small monetary incentive.

Different representations for presenting a shared mobility system to a reinforcement learning agent will have varying performance and a well-structured representation can show significant improvement. In experimentation bikeshare trip-data from Washington D.C.'s Capital Bikeshare system between 2015 and 2019 was utilized to produce data-driven simulations. Proposed representations were evaluated for performance within these scenarios as to how effectively they can promote the service level of a system.

All evaluated representations outperform the no agent baseline, showing that they do provide a benefit to a system's service level. A lower dimensional state representation such as only providing the supply level per region shows strong performance, as seen in S-A and S-A+, as the supply level is the most crucial information for an agent to determine an incentive scheme. Representations with a high dimensional state representation, as seen in SADUE-A, perform poorly as an agent is unable to determine crucial information within the representation. Further, an improved state representation that includes predicted activity in the following time slot, seen in the representation SA'D'-A+, shows a slight improvement over just supply information.

The action mechanism A+ outperforms both representations with the too general action mechanism signified A and the too complex action mechanism signified as A+D+. The A action mechanism does not provide the agent with enough control over user decisions, as it simply provides an offer to move in any direction arbitrarily. Whereas the A+ mechanism allows an agent to more purposely incentivize users to move toward or avoid specific regions. The A+D+ action mechanism showed a significant reduction in performance compared to other representations until supplied a very large budget. This suggests that this action mechanism is difficult to effectively utilize due to its larger budget consumption. Overall, the representation SA'D'-A+ performs the best of the evaluated representations with S-A and S-A+ closely behind. SA'D'-A+'s strong performance is due to its use of a predictive model and direct information for estimating user behavior coupled with an action mechanism that gives the agent adequate ability to incentivize the user to slightly change their behavior in a controlled way.

Lastly, in experimentation addressing varying environmental constraints, such as user demand and available supply, it was observed that agents are able to greatly improve service level as the system supply is reduced. This can improve operating costs through reduced maintenance and equipment expenses, as well as improve brand reputation through less sidewalk clutter. However, in experimentation agents show more difficulty in adapting to an increase in user demand. In all cases of environmental constraints agents show an improvement in service level versus no agent.

8.2 Future Work

8.2.1 Hierarchical Approaches

In experimentation, many concepts from the literature were used or inspired the creation of the tested representations. However, hierarchical reinforcement learning is a popular approach in the literature that was not evaluated [6] [34] [35]. It seems that a global-localized model would be able to train in a more sample-efficient manner while also reducing implementation complexity. A comparison against this methodology would be interesting to observe if this assumption is true. Further, some components of hierarchical reinforcement learning could be integrated into the tested approaches. An effective approach may be to train a global-localized model until it reaches convergence, then deploying copies of that model to each individual region where a hierarchical approach can be applied. This approach can then fit each model to its own region while also keeping all models similar in performance.

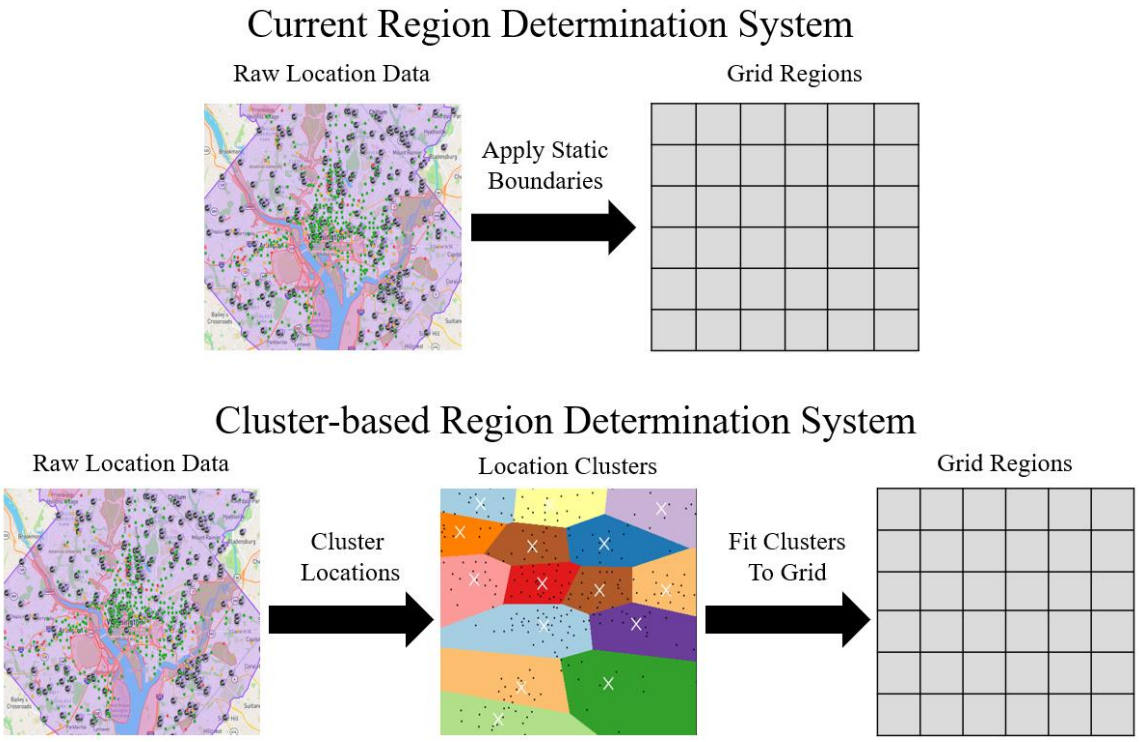
8.2.2 Per User Incentivization

Instead of using static hourly region-based incentives a more customized implementation can be employed where individual users are observed along with the current system state to be provided an individualized incentive. We conducted some preliminary exploration of this avenue and found that it presented a variety of unique properties that separated it as an approach from the hourly-based regional incentive structure. It presents the challenge of how to best present relevant information to the agent, it is critical that key pieces of information such as the user's position is not lost in a large state space with trivial information such as the supply of a region on the other side of the system.

8.2.3 Cluster-Based Regions

Figure 20

Cluster-Based Region System



The use of clustering to determine system regional boundaries based on user activity or points of interest is likely to produce regions that better capture a representation of the system. Currently, we utilize equal-sized grided regions, however in our use case there exists many data repositories detailing tourist points of interest in Washington D.C. . Grided regions were used in experimentation due to computational efficiency and simplicity, however there are certainly methods for mapping unevenly shaped regions into the modeled environment.

The idea of cluster-based regions can be further extended to incorporate traffic networks. In one case the traffic network in which those using the bikeshare system can engage in can be represented as a graph to encapsulate available routes and capture traffic on said routes. Much of the current literature relies upon the usage of straight-line distances in calculating user costs and trip durations, however cities are typically very constrained mobility environments with highly variable traffic patterns and delays.

References

[1] L. P. Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research,* no. 4, pp. 237-285, 1996.

[2] OpenAI, "Dota 2 with Large Scale Deep Reinforcement Learning," 2019.

[3] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps and D. Silver, "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning," *Nature,* no. 575, pp. 350-354, 2019.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. v. d. Driessche, T. Graepel and D. Hassabis, "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature,* no. 529, pp. 484-489, 2016.

[5] L. An, C. Ren, Z. Gu, W. Yuexuan and Y. Gao, "Rebalancing the Car-Sharing System: A Reinforcement Learning Method," in *IEEE International Conference on Data Science in Cyberspace (DSC*, Hong Kong, 2019.

[6] L. Pan, Q. Cai, Z. Fang, P. Tang and L. Huang, "A deep reinforcement learning framework for rebalancing dockless bike sharing systems," in *Proceedings of the AAAI conference on artificial intelligence*, Honolulu, Hawaii, USA, 2019.

[7] W. G. Hatcher and W. Yu, "A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends," *IEEE Access,* vol. 6, pp. 24411-24432, 2018.

[8] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Gues, L. Edward, D. Hassabis, T. Graepel, T. Lillicrap and D. Silver, "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model," *Nature,* no. 588, pp. 604-609, 2020.

[9] J. Pfrommer, J. Warrington, G. Schildbach and M. Morari, "Dynamic vehicle redistribution and online price incentives in shared mobility systems," *IEEE Transactions on Intelligent Transportation Systems,* pp. 1567-1578, 2014.

[10] A. Singla, M. Santoni, G. Bartók, P. Mukerji, M. Meenen and A. Krause, "Incentivizing users for balancing bike sharing systems," in *Proceedingsof the AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, 2015.

[11] S. Ghosh and P. Varakantham, "Incentivizing the use of bike trailers for dynamic repositioning in bike sharing systems," in *Proceedings of the International Conference on Automated Planning and Scheduling*, Nancy, France, 2017.

[12] H. Lv, C. Zhang, Z. Zheng, T. Luo, F. Wu and G. Chen, "Mechanism Design with Predicted Task Revenue for Bike Sharing Systems.," in *Conference on Artificial Intelligence (AAAI)*, New York, New York, USA, 2020.

[13] Y. Chabchoub, R. Sibai and C. Fricker, "Bike Sharing Systems: a New Incentive Rebalancing Method Based on Spatial Outliers Detection," *International Journal of Space-based and Situated Computing*, vol. 9, no. 2, pp. 99-108, 2019.

[14] F. Chiariotti, C. Pielli, A. Zanella and M. Zorzi, "A Bike-sharing Optimization Framework Combining Dynamic Rebalancing and User Incentives," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 14, no. 3, pp. 1-30, 2020.

[15] Z. Li, J. Zhang, J. Gan, P. Lu, Z. Gao and W. Kong, "Large-scale trip planning for bike-sharing systems," *Pervasive and Mobile Computing*, vol. 54, pp. 16-28, 2019.

[16] D. Tomaras, V. Kalogeraki, T. Liebig and D. Gunopulos, "Crowd-based Ecofriendly Trip Planning," in *19th IEEE International Conference on Mobile Data Management (MDM)*, Aalborg, Denmark, 2018.

[17] Y. Daun and J. Wu, "Optimizing Rebalance Scheme for Dock-less Bike Sharing Systems with Adapative User Incentive," in *IEEE International Conference on Mobile Data Management (MDM*, Hong Kong, 2019.

[18] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction 2nd Edition, Cambridge, Massachusetts: The MIT Press, 2018.

[19] R. E. Bellman, Dynamic Programming, Princeton: Princeton University Press, 1957.

[20] R. E. Bellman, "A Markov Decision Process," *Jounral of Mathematics and Mechanics*, pp. 679-684, 1957.

[21] E. L. Thorndike, Animal Intelligence: Experimental Studies, Macmillan Press, 1911.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *Neural Information Processing Systems (NIPS)*, Lake Tahoe, USA, 2013.

[23] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in *Internaional Conference on Machine Learning (ICML)*, Beijing, China, 2014.

[24] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu and N. d. Freitas, "Sample Efficient Actor-Critic with Experience Replay," in *Internation Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[25] T. Haarnoja, A. Zhou, P. Abbeel and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018.

[26] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in *International Conference on Machine Learning (ICML)*, Beijing, China, 2014.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," in *International Conference on Machine Learning (ICML)*, New York City, USA, 2016.

[28] J. Schulman, S. Levine, P. Mortiz, M. l. Jordan and P. Abbeel, "Trust Region Policy Optimization," 2015.

[29] Y. Wu, E. Mansimov, S. Liao, R. Grosse and J. Ba, "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation," 2017.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," 2017.

[31] S.-S. Ho, M. Schofield and N. Wang, "Learning Incentivization Strategy for Resource Rebalancing in Shared Services with a Budget Constraint," *Journal of Applied and Numerical Optimization*, vol. 3, no. 1, pp. 105-114, 2021.

[32] Capital Bikeshare, "System Data," Capital Bikeshare, [Online]. Available: https://www.capitalbikeshare.com/system-data. [Accessed 23 Febuary 2021].

[33] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[34] A. Vezhnevets, V. Mnih, J. Agapiou, S. Osindero, A. Graves, O. Vinyals and K. Kavukcuoglu, "Strategic Attentive Writer for Learning Macro-Actions," in *Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.

[35] O. Nachum, S. Gu, H. Lee and S. Levine, "Data-Efficient Hierarchical Reinforcement Learning," in *Neural Information Processing Systems (NIPS)*, Montreal, Canada, 2018.