Eötvös Loránd University

Faculty of Informatics

Dept. of Software Technology And Methodology

Credit Assignment in Traffic Signal Control

Supervisor:
Zoltán Istenes
Associate Professor

Author:
Diab Radwan
Computer Science MSc

Budapest, 2021

Topic of the Thesis:

Traffic Signal Control is the task of controlling traffic to reduce congestion and minimize travel times. Traffic control is a complex task especially in large scale traffic scenarios in which there is no optimal solution; in such cases multi-agent reinforcement learning algorithms have been popularly used to solve this problem. In these algorithms local controllers are considered agents that control traffic going through their junction. Agents learn to make their decisions based on the observed local state of traffic; however, there is still the need for coordinating with other agents, even if all agents make decisions that are locally optimal they still might not be optimal or efficient globally. Hence coordination is one of the major points for multi-agent reinforcement learning. Another concern is credit assignment, given a local reward estimate after taking a set of actions it is difficult to determine the contribution of each of these actions towards that local reward, in other words in case of a positive reward we can not attribute that solely to the action of the local agent, the reward is influenced largely by the actions of other agents. The purpose of this thesis is to address the problem of credit assignment in traffic signal control tasks by applying a modification to Generalized Advantage Estimation that utilized the concept of Afterstates and abuses the fact that in traffic environment we know the immediate result of taking a particular action. As for the model a recent architecture called Deep Implicit Coordination Graphs is used to address the coordination issue. The thesis uses simulated traffic over a realistic road network. The thesis aims to analyze and compare the results and limitations of this approach to the results of other multiagent reinforcement learning algorithms applied in this domain.

Contents

1 Introduction **3**

1.1 Traffic Congestion 3

1.2 Traffic Signal Control 3

1.3 Reinforcement Learning based Traffic Signal Control 4

1.4 Implemented Approach 6

1.5 Thesis Structure 6

2 Background **7**

2.1 Reinforcement Learning 7

2.2 Reinforcement Learning with Function Approximators 8

2.3 Policy Optimization 9

2.3.1 Proximal Policy Optimization 9

2.3.2 Generalized Advantage Estimation 10

2.4 Deep Implicit Coordination Graphs for Baseline Approximation . . . 10

2.4.1 Decentralized Policy 11

2.4.2 Centralized Policy 12

2.5 Reinforcement Learning in Traffic Signal Control 12

3 Methodology **14**

3.1 Architecture 14

3.2 State and Reward 15

3.3 Advantage Estimation Using Afterstates 15

3.4 Artificial Episode Termination 16

3.5 Training Algorithm 16

3.6 Technologies 18

3.7 Datasets 18

3.8 Experiments 19
3.9 Evaluation and Metrics 19
3.10 Implementation Code 20

4 Results **21**
4.1 Synthesized 4-by-4 Road Network 21
4.2 Effects of using Deep Implicit Coordination Graphs and Modified GAE 24

5 Conclusion **27**
5.1 Advantages 27
5.2 Limitations 28
5.3 Future Work 28

A Hyperparameters **30**
A.1 Synthesized 4-by-4 dataset hyperparameters 30

Bibliography **32**

List of Figures **35**

List of Tables **36**

Chapter 1

Introduction

1.1 Traffic Congestion

Traffic congestion is a major issue in highly populated and developing cities. According to a traffic study performed by data and analytics company Inrix [1]; each driver in New York lost, on average, 100 hours due to congestion in 2020 despite the COVID-19 restrictions and lockdown. In fact the number dropped from 140 hours in 2019. The time lost is calculated as the difference between the travel time at peak traffic and travel time at free flow conditions.

According to the same study [1] the costs to the city of New York due to congestion in 2020 are estimated to be $7.7B which is a lower number than previous years due to the pandemic. Similar numbers appear in the report for some of the large cities that were part of the study such as London, Paris, Rome and Philadelphia. Even relatively smaller cities such as Bucharest and Zagreb suffered significant time and fuel losses due to congestion.

1.2 Traffic Signal Control

Solutions that leverage technology advancement are being developed and employed to address the growing problem of traffic congestion, one of these solutions is traffic light control.

Traffic lights are signals deployed at road intersections and signal crossings to control the flow of traffic and ensure the safety of drivers and pedestrians. In their

simplest form traffic lights follow fixed time plans but more elaborate traffic control systems that take into account nearby signals, pedestrian buttons and traffic volumes; these systems coordinate and adapt to changing conditions in order to reduce congestion and provide shorter waiting and travel times.

Recently research presenting AI-based traffic light control systems emerged and achieved impressive results [2]. Some of the most successful AI algorithms for traffic light controls are based on Reinforcement Learning.

1.3 Reinforcement Learning based Traffic Signal Control

Reinforcement Learning is a machine learning paradigm in which an agent learns a policy to interact with its environment in order to maximize a reward signal [3]; the agent observes the environment possibly partially and the policy is learned by trial and error. Recent papers that combine reinforcement learning with deep reinforcement learning technologies have achieved significant success and made Deep Reinforcement Learning a popular choice for decision making and control tasks.

For any reinforcement learning task a critical aspect is the design of the state and the reward. The state being what the agent observes and the reward is the score it tries to maximize; for traffic light control the objective is to minimize the average travel time of all vehicles across the entire road network; this objective is difficult to minimize directly so alternative measures are used and the reward is designed based on traffic theory. Similarly, a multitude of choices exist for the state, ranging from camera images of the intersection to simply the number of vehicles on each lane.

Another critical aspect of reinforcement learning is its trial and error nature; essentially reinforcement learning agents train their policy by exploring the state space. They consider how different actions in various states affect the cumulative reward and based on that the probability of performing said action is increased or decreased. This makes training these agents in real environments not feasible as it would take a long time during which it would extremely hinder the operation of the road network.

To resolve this issue traffic simulators are used to train reinforcement learning agents, these simulators attempt to mimic real traffic environments as closely as possible; providing a method to safely and quickly train agents for traffic signal control. These simulators allow users to design their own road networks and traffic flows which may be real or synthetic. This thesis uses a recently developed simulator called CityFlow [4]. CityFlow is designed with reinforcement learning in mind and is focused on scalability and simulating large road networks while maintaining its performance.

Research shows that single-agent reinforcement learning methods succeed in traffic light control tasks and achieves impressive results for small-sized road networks. For larger road networks with many intersections single-agent reinforcement learning either can not scale to that size or fail to achieve acceptable results, this is due to the large size of the joint action space and joint state space; thus the need for Multi-Agent Reinforcement Learning.

Multi-Agent Reinforcement Learning in the context of traffic signal control trains an independent agent to control an intersection or a subset of intersections thus dividing the road network; this has the effect of keeping the state and action spaces small and allows to scale reinforcement learning to large road networks of many intersections. Using Multi-Agent Reinforcement learning introduces new difficulties; for example what information has to be communicated between agents to make their decisions and which agents need to communicate, and how can agents coordinate their actions to improve performance. Another example is the credit assignment problem which is critical even for single-agent reinforcement learning but even more so in multi-agent settings; the core problem to address is how to attribute an increase or decrease in rewards to a particular action or set of actions and in the case of multi-agent settings attributing the changes in rewards to an action of a particular agent.

1.4 Implemented Approach

In this thesis a recent architecture called Deep Implicit Coordination Graphs is applied to the traffic signal control problem, this architecture uses Graph Convolution Networks and attention mechanisms to dynamically learn a coordination graph between agents. This architecture attempts to address the coordination issue in Multi-Agent Reinforcement Learning. In this work an actor-critic algorithm is implemented where the critic is implemented using the DICG architecture to approximate a value function.

The agent is trained using Proximal Policy Optimization which is one of the most popular policy gradient methods and advantage estimation is done using Generalized Advantage Estimation.

Furthermore, a new approach to address the credit assignment problem for traffic signal control is proposed by this thesis. The concept of Afterstates is utilized and a traffic signal control specific modification to Generalized Advantage Estimation is proposed that significantly improves results. The approach is evaluated and compared to vanilla GAE and other recent research on the topic using synthetic road networks.

1.5 Thesis Structure

Chapter 2 is concerned with providing a background and basic literature review of related works. Next, chapter 3 discusses in detail the general setting, the proposed approach and the experiments conducted. After that chapter 4 discusses the results and analyses the effects on performance of the proposed advantage estimation approach, and it compares the results to recent research in the field. Finally chapter 5 discusses final conclusions, limitations and future work.

Appendix A contains a more detailed description of the training procedure and hyperparameter values.

Chapter 2

Background

This Chapter provides a background and sets the context of this thesis. It introduces formally the framework of Reinforcement Learning and the algorithms applied. Finally it reviews related works in the field of traffic signal control.

2.1 Reinforcement Learning

According to Sutton and Barto (2018) [3], Reinforcement Learning is a paradigm of machine learning in which an agent learns to interact with an environment in order to maximize a numerical reward signal.

Reinforcement Learning problems that satisfy the Markov property are usually posed as a Markov Decision Process MDP [3]. An MDP consists of: the state space S, the action space A, and the environment dynamics:

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{2.1}$$

At each timestep t the agent is presented a state $S_t \in S$ and selects an action $A_t \in A$. After taking the action the agent receives a reward R_t and the next state S_{t+1} from the environment based on the dynamics function p. The objective is to maximize the total reward the agent receives from the environment $\Sigma_{t=0}^{\infty} r_t$. In this paper we focus on Model-Free Reinforcement Learning in which the agent has no access to and does not attempt to directly learn the environment dynamics.

In order to select actions at each timestep the agent learns a policy π which is a mapping from the perceived state to an action to be taken. Usually the policy is

stochastic and it is defined as a probability distribution from states to actions [3]. Where $\pi(a|s)$ is the probability of taking action a when the agent receives the state s.

Most Reinforcement Learning algorithms estimate the expected sum of discounted rewards given a particular state s and following a specific policy π, this estimate is called the value function:

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \tag{2.2}$$

Where γ is the discount-rate which is selected to be a value between 0 and 1 and it has the effect, depending on the value selected, of making the agent value immediate and short-term rewards more than long-term future rewards. In the case of $\gamma < 1$ it keeps the discounted sum of rewards finite if the reward sequence is bounded.

Another common function is the action-value function of a state, it estimates the value of taking an action a when in state s and then following a policy π. It is defined as:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right] \tag{2.3}$$

A brand of Reinforcement learning algorithms attempt to estimate the value or the action-value function of a policy and then proceed to improve the policy based on the Policy Improvement Theorem [5]. Most notably Q-Learning [6] methods attempt to learn the optimal action-value function q^*, the action-value function of the optimal policy π^*.

2.2 Reinforcement Learning with Function Approximators

For small and finite state spaces these value functions can be kept in a tabular fashion and a rich set of tabular algorithms can be used to estimate these functions. Tabular methods can not be used for large and possibly infinite state spaces, in such cases parameterized function approximators are trained to fit the value function. Some of the tabular algorithms were extended to be used with function approximators and new algorithms were proposed.

Historically there has been many architectures used as function approximators, for example Teasuro's TD-Gammon used an Artificial Neural Network applied to a set of hand-crafted features to learn backgammon [7]. In 2013 Mnih et.al [8] introduced a Deep Reinforcement Learning agent that learned to play Atari games using raw frames as the observations, a Convolutional Neural Network [9] was used as an approximator for the action-value function.

2.3 Policy Optimization

A family of Reinforcement Learning methods directly represent the policy to be followed $\pi_\theta(a|s)$ as a function parameterized by θ and then θ is optimized either directly or indirectly. Still these methods require an estimate of the value function $v_\pi(s)$ in order to compute the updates to the policy. Actor-Critic methods are an example of this family of algorithms where the critic is usually an approximation of the value function and the actor is a parameterized policy [3].

Policy Optimization methods are on-policy methods, meaning updates are performed using data generated by actions taken by the latest policy whereas methods belonging to Q-Learning are off-policy [3], meaning updates can be done using data generated at any time of the training, making these algorithms more sample efficient. However, Policy Optimization methods are shown to be more stable than Q-Learning methods. .

2.3.1 Proximal Policy Optimization

One of the most popular Policy Optimization methods is Proximal Policy Optimization [10] where an advantage estimate $\hat{A}_t$ is calculated using Generalized Advantage Estimation and with some trained baseline. The PPO loss used to train the policy is described by the equation

$$\mathrm{L}_{actor} = \mathbb{E}_t \left[min\Big(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t, clip\big(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \big) \hat{A}_t \Big) + c_2 S[\pi_\theta(s_t)] \right] \quad (2.4)$$

where ϵ is the clipping parameter, S is an entropy bonus and c_2 is a coefficient. The policy can be a single network that is shared across all agents or it can be a separate network for each agent.

2.3.2 Generalized Advantage Estimation

The advantage function $\hat{A}_t$ can be estimated using Generalized Advantage Estimation [11], this method estimates the advantage at each time step t as

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^{V_\theta} \tag{2.5}$$

where γ is the discount factor, λ is the GAE parameter and $\delta_t^{V_\theta}$ is the TD residual error estimated using the baseline as

$$\delta_t^{V_\theta} = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t) \tag{2.6}$$

2.4 Deep Implicit Coordination Graphs for Baseline Approximation

Deep Implicit Coordination Graphs (DICG) is a Multi-Agent Reinforcement Learning architecture proposed by Li et al. in 2021 [12]. This architecture allows for learning a dynamic and state dependent coordination graph structure that is used in a Graph Convolutional Network [13] to build observation embeddings.

In this method for each timestep the observations o_i^t of n agents are put into an Encoder which is a multilayer perceptron where the final layer has d units which is the desired size of the embedding. The result is an array containing an embedding for each agent observation $E_t \in \mathcal{R}^{n \times d}$.

Following that a self-attention mechanism [14] is used to calculate an adjacency matrix for the following Graph Convolutional Network. The mechanism is parameterized by an array of trainable weights $W_a \in \mathcal{R}^{d \times d}$. The attention score between two embeddings is calculated as

$$a_{ij}^t = {e_j^t}^T W_a e_i^t \tag{2.7}$$

These scores are used to populate the adjacency matrix M_t of size $n \times n$ where

$$m_{ij}^t = \frac{\exp(a_{ij}^t)}{\sum_{k=1}^{n} \exp(a_{ik}^t)} \tag{2.8}$$

This adjacency matrix M^t and the embeddings matrix E^t are put into a Graph Convolutional Network [13]. The GCN consists of several layers and each layer l has a set of trainable weights $W_c^{(l)}$ and an activation function σ. Denoting $H^{(0)} = E^t$ as the input of the first GCN layer we can calculate the output of layer l as

$$H^{(l+1)} = \sigma(M^t H^{(l)} W_c^{(l)}) \tag{2.9}$$

The output of the last layer, denoted as $\tilde{E}_t$, contains new embeddings which are the result of the integration and message passing preformed by the GCN between the original embeddings and based on the attention scores.

Li et al. further propose to add the starting embeddings E^t to the resulting embeddings $\tilde{E}_t$ as a form of residual connection.

Next each embedding is passed to a multilayer perceptron denoted as MLP which approximates the value function of the current policy $V_\pi(s_t)$ based on these embeddings.

This architecture is trained to minimize the Mean Squared Error between the output and the discounted sum of rewards of each agent.

$$L_{baseline} = \mathbb{E}\left[\left(MLP(\tilde{E}_t) - \sum_{l=0}^{\infty} \gamma^l r_{t+l}\right)^2\right] \tag{2.10}$$

In the paper Li et al. propose two architectures, centralized-training-centralized-execution and centralized-training-decentralized-execution.

2.4.1 Decentralized Policy

The policy can be a multilayer perceptron which takes an agent observation and outputs logits which are used to parametrize a categorical distribution. Since the policy accepts raw observations as inputs, it is decentralized and can run without communication between the agents and does not require the baseline to act. The paper proposes training the networks using actor-critic methods.

2.4.2 Centralized Policy

In the case of a centralized policy where full communication is allowed between agents, the policy network operates using the embeddings $\tilde{E}$ as input. Since it now shares parameters with the baseline they can no longer be trained separately and should be trained to optimize a shared objective such as

$$L = c_1 L_{baseline} + L_{policy} \tag{2.11}$$

where c_1 is a coefficient.

In this approach the full architecture is needed to take actions and the observations of all agents are used.

2.5 Reinforcement Learning in Traffic Signal Control

In the task of Traffic Signal Control each intersection controls a set of traffic lights by choosing which phase it activates. A phase is a combination of movement signals, where a green signal indicate that the movement is allowed and a red signal indicates that the movement is prohibited [2]. The number of traffic lights controlled depends on the number of lanes in the intersection.

In large road-networks it becomes difficult to scale a single reinforcement learning agent to control the entire network of intersections, as such the problem is treated as a multi-agent problem where each agent controls a single intersection.

The observation of each agent at timestep t denoted as o_t consists of the number of entering and exiting cars on each traffic lane and the intersection's currently active phase p_t.

Each agent selects a phase p_t as its action at time step t denoted as a_t. This phase remains active for a minimum number of timesteps Δt which is the action interval.

The objective of traffic signal control tasks is to minimize the average travel time across all vehicles entering the road-network. However, minimizing that objective directly is too difficult, hence the local reward at timestep t denoted as R_t is the sum of the number of vehicles waiting on each lane in the intersection. Since agents

take an action every Δt timesteps, the immediate reward presented to the agent is the mean reward across all Δt timesteps.

The design of the state representation and the reward function are open issues, some works have found success and improved performance by incorporating traffic theory concepts such as max-pressure into the design of the state representation and reward function [2] [15], however this is not the focus of this work. Wei et al. provide a comprehensive survey on traffic signal control methods [16].

Using the architecture of Deep Implicit Coordination Graphs as the baseline for generalized advantage estimation allows each intersection to utilize the observations of other agents to estimate the value of its current state. However this does not solve the credit assignment problem. Given a state-action pair and a reward that is advantageous to the expected value of the state, the problem is whether credit for this advantage goes to the local agent's action or to actions of other agents perhaps at neighbouring intersections.

Chapter 3

Methodology

This chapter describes the approach applied in this thesis to solve the traffic signal control problem. Proximal Policy Optimization was used to train the agent and the advantage function was estimated using a modified Generalized Advantage Estimation that leverages a special attribute of the environment. Furthermore the baseline for advantage estimation was approximated using the Deep Implicit Coordination Graphs architecture which allows for improved coordination between agents for estimating the baseline for GAE.

3.1 Architecture

The thesis applies the Deep Implicit Coordination Graphs [12] centralized-training-decentralized-execution architecture that is described in section 2.4, a multilayer perceptron is used as the policy and its parameters are shared across all agents. The architecture is trained in an actor-critic style as described in the original paper and the PPO method described in section 2.3.1 is used to train the policy. The thesis proposes using Afterstates and a modified version of General Advantage Estimation 2.3.2 that improves credit assignment between the different agents. The Afterstates approach and the advantage estimation method are described in section 3.3.

3.2 State and Reward

As discussed in the previous chapter the literature has produced a variety of choices when it comes to the design of the state and reward for the agents. In this thesis the focus is evaluating the architecture and algorithm itself and not the design choice for the state or reward functions. In this work the observation of an agent consists of the number of vehicles present in each lane of the intersection and the current active traffic phase. The reward in each timestep is the negative sum of the waiting time of the vehicles currently in the intersection.

3.3 Advantage Estimation Using Afterstates

In traffic signal control tasks we have a unique situation where the instantaneous next state of the agent is known based on the previous state and its chosen action.

Since the observation of the agent consists of a vector of the number of vehicles entering and exiting the intersection on each lane denoted as $\mathbf{v_t}$ and the current active phase p_t. Upon taking an action and selecting the next phase the state of the intersection changes to have p_{t+1} as the phase and $\mathbf{v_{t+1}} = \mathbf{v_t}$.

This unique attribute of the task allows for using afterstates instead of states, that is, the baseline is trained to estimate the value of the state that immediately follows the action's execution. Effectively using $\hat{s}_t = (\mathbf{v}_t, p_{t+1})$ for value estimation and the original states s_t for decision making.

In order to address the credit assignment problem, we must be able to calculate an advantage for agent i which is largely attributed to its own action at a particular timestep a_t^i. Using the concept of afterstates can help calculate such advantage.

In equation 3.2 the advantage is calculated off of the estimated value of the state, this estimate is calculated using the DICG architecture which accepts all observations as input. By applying the concept of afterstates the advantage can be calculated off of the afterstate $V(\hat{s}_t)$ instead. However, at this point we can modify the afterstate $\hat{s}_t$ by setting the phase of agent i to its previous value

$$\hat{S}_t^{\neg a_t^i} = (\mathbf{V}_t, p_{t+1}^0, p_{t+1}^1, ..., p_t^i, ...p_{t+1}^n) \tag{3.1}$$

effectively disregarding the action taken by agent i. Calculating the advantage for agent i by applying equation 3.2 to the afterstate value of $\hat{s}_t^{\neg a_t^i}$ instead of S_t results in advantage values that are attributed to the action taken by agent i. Following this approach, the advantage is calculated as follows:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^{V_\theta} \tag{3.2}$$

Where the td error is now calculated as:

$$\delta_t^{V_\theta} = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(\hat{s}_t^{\neg a_t^i}) \tag{3.3}$$

By repeating the same process for each agent we can estimate advantages that credit the agents' local actions more accurately.

3.4 Artificial Episode Termination

Traffic signal control is a continuous reinforcement learning task, meaning there are no episodes and the execution goes on forever. However, to be able to train an algorithm a practical limit must be set and execution must be done in episodes. In this type of tasks it creates a problem of an artificial termination where the discounted sum of rewards becomes closer to zero towards the end of the episode, and in our case of negative rewards it leads the agents to believe that their actions led to the end of the episode and thus these actions are overvalued. Pardo et al. wrote a paper on this issue and proposed some solutions [17]. However, in our case we simply extend the episode by $10 * \Delta t$ timesteps and discard the data collected during the extended period, this has the effect of smoothing the artificial termination out.

3.5 Training Algorithm

The training procedure follows the algorithm described in algorithm 1. Where:

- Δt is the action interval
- N is the number of agents

- P is the number of desired episodes
- o_t^i, a_t^i, r_t^i and v_t^i are the observation, action, reward and value for agent i at timestep t respectively.

Training is performed in epochs of randomly sampled mini batches.

Algorithm 1 Training procedure

Initialize critic network based on the DICG architecture V_θ and actor MLP network A_ϕ
for *episode* $\leftarrow$ 0 to P **do**
  for $t \leftarrow$ 0 to M=$T/\Delta t$ **do**
    for agent $i \leftarrow$ 0 to N **do**
      sample action a_t^i from distribution $A_\phi(o_t^i)$
    end for
    for $k \leftarrow$ 0 to Δt **do**
      perform action a_t^i for each agent i
    end for
    $r_t^i \leftarrow$ mean of rewards received by agent i during the period Δt
    collect a_t^i, o_t^i and r_t^i
  end for
  compute v_t^i as described in equation 2.2
  minimize $\sum_{t=0}^{M} \left(V_\theta^i(o_t) - v_t^i\right)^2$ w.r.t. θ with specified epochs and mini batch size

  calculate advantage for each action a_i^t using equation 3.2
  minimize PPO loss (2.3.1) w.r.t. ϕ with specified epochs and mini batch size
end for
return A_ϕ

The experiments performed differ from the pseudo code in either the architecture of the critic or the method of calculating the advantage estimates.

3.6 Technologies

All the experiments preformed in this thesis were implemented using the Python programming language. The model and learning loop were implemented using TensorFlow 2. The traffic simulations were performed using the CityFlow [4] simulator. In addition, an open-source Gym [18] compatible environment for CityFlow was used [19]. The following is a table containing the exact versions of all the tools and technologies that were used:

Name	Version
Python	3.8
TensorFlow	2.4.1
TensorBoard	2.4.1
OpenAi Gym	0.18.0
CityFlow	0.1

Table 3.1: Table listing the technologies used.

3.7 Datasets

The experiments performed using the CityFlow simulator are trained and evaluated over a synthetic dataset that is used for evaluation in the most recent related paper [15] that utilized CityFlow. The results are used to compare with this paper. The dataset consist of a road network with varying traffic flows.

The dataset described in [15] consists of a road network that is a 4-by-4 network and each road segment connecting two intersections is 300 meters long. Multiple traffic flows exist for the road network having either 0.388 or 0.416 average arrival rate per second with either 0.3 or 0.6 variance giving 4 possible combinations.

This dataset was chosen as it was the largest road network that has reported results and is not too large compared to our available time and resources.

A large dataset based on an area of New York consisting of hunderds of intersections is described and used in [15] and [20], however due to the lack of proportional computational resources this road network was not used.

3.8 Experiments

A set of agents is trained and evaluated for the synthesized 4-by-4 dataset. In all experiments the action interval Δt was fixed to 20 to keep the experiments running time low. All agents are trained using the DICG 2.4 architecture as a critic and PPO 2.3.1 to train the policy network.

A set of agents are trained and evaluated using the vanilla GAE for advantage estimation 2.3.2 and a different set of agents using the modified GAE proposed by this thesis 3.3. The purpose is to analyse and compare the results the two approaches.

Furthermore to analyse the effects of using DICG as a baseline, a set of agents is trained using a simple Multi-Layer perception as the critic.

All agents were trained for 200 episodes and each episode simulates traffic for 30 minutes. Our best performing agent is compared to other methods reported in [15].

3.9 Evaluation and Metrics

The performance of a set of agents is evaluated each episode by calculating the average travel time of all vehicles across all intersections for the entire length of the episode, during these episodes a stochastic policy is used. After an agent has fully finished training, the same metric is calculated for the greedy policy. The greedy policy evaluation is used to compare with other works whereas the former mentioned results are visualized to compare between the different approaches used in this work, namely using the DICG architecture compared to a simple MLP baseline and using GAE compared to the proposed modified GAE.

It is worth noting that the average travel time is calculated using CityFlow's API's whereas related works being compared [20] [15] use a different method for calculation which does not account for the time vehicles spent on the last lane before leaving the network and the time spent passing through the intersections.

In addition to the average travel time, training related metrics are collected and logged after each episode, these include the mean training loss for the actor and the critic and the sum of total rewards during the episode.

3.10 Implementation Code

The code implementing the described approach is available as a GitHub repository at the following link: github.com/r7sy/Traffic-Signal-Control.

The repository also includes the CityFlow compatible road network and traffic flow files used which are taken from [15] [20]. It also includes TensorBoard event files containing logged metrics of the experiments.

Chapter 4

Results

This chapter contains the results of the experiments listed in section 3.8 in addition to visualizations to compare the different approaches implemented in this thesis. In addition it presents a table containing a comparison between the performance of our best agent to the results of other methods reported in the literature.

4.1 Synthesized 4-by-4 Road Network

The following table contains the average travel time achieved by our and several other methods as reported by [15] for the synthesized 16 intersection dataset described in section 3.7, the average travel time is calculated across all vehicles in one 30 minutes long episode. Our agent was trained for 200 episodes and the hyperparameters used are listed in Appendix A.

Method	Average Travel Time			
	Config 1	Config 2	Config 3	Config 4
MaxPressure	361.17	402.72	360.05	406.45
PressLight	354.94	353.46	348.21	398.85
FRAP	340.44	298.55	361.36	598.52
MPLight	309.33	262.5	281.34	353.13
DICG + Modified GAE	257.5	301.6	289.5	347.6

Table 4.1: Table containing results over the synthesized 4by4 dataset.

Here the configurations listed are detailed as:

- Config 1: 0.388 arrival rate with 0.3 variance
- Config 2: 0.388 arrival rate with 0.7 variance
- Config 3: 0.416 arrival rate with 0.3 variance
- Config 4: 0.416 arrival rate with 0.7 variance

The results show that our method outperforms other compared methods for configurations 1 and 4 whereas for configuration 3 it performs slightly worse and for configuration 2 it performs significantly worse. The results show a significant decline in performance when comparing config 1 and 2 which differ in the variance of the traffic arrival rate, thus the method responds poorly to high traffic variance compared to low variance settings. The performance over configuration 4 is also significantly worse when compared to configuration 3 but still it responded better to the increase in variance when compared to other methods.

Figure 4.2 shows the average travel time of all vehicles for each training episode and for all the configurations available.

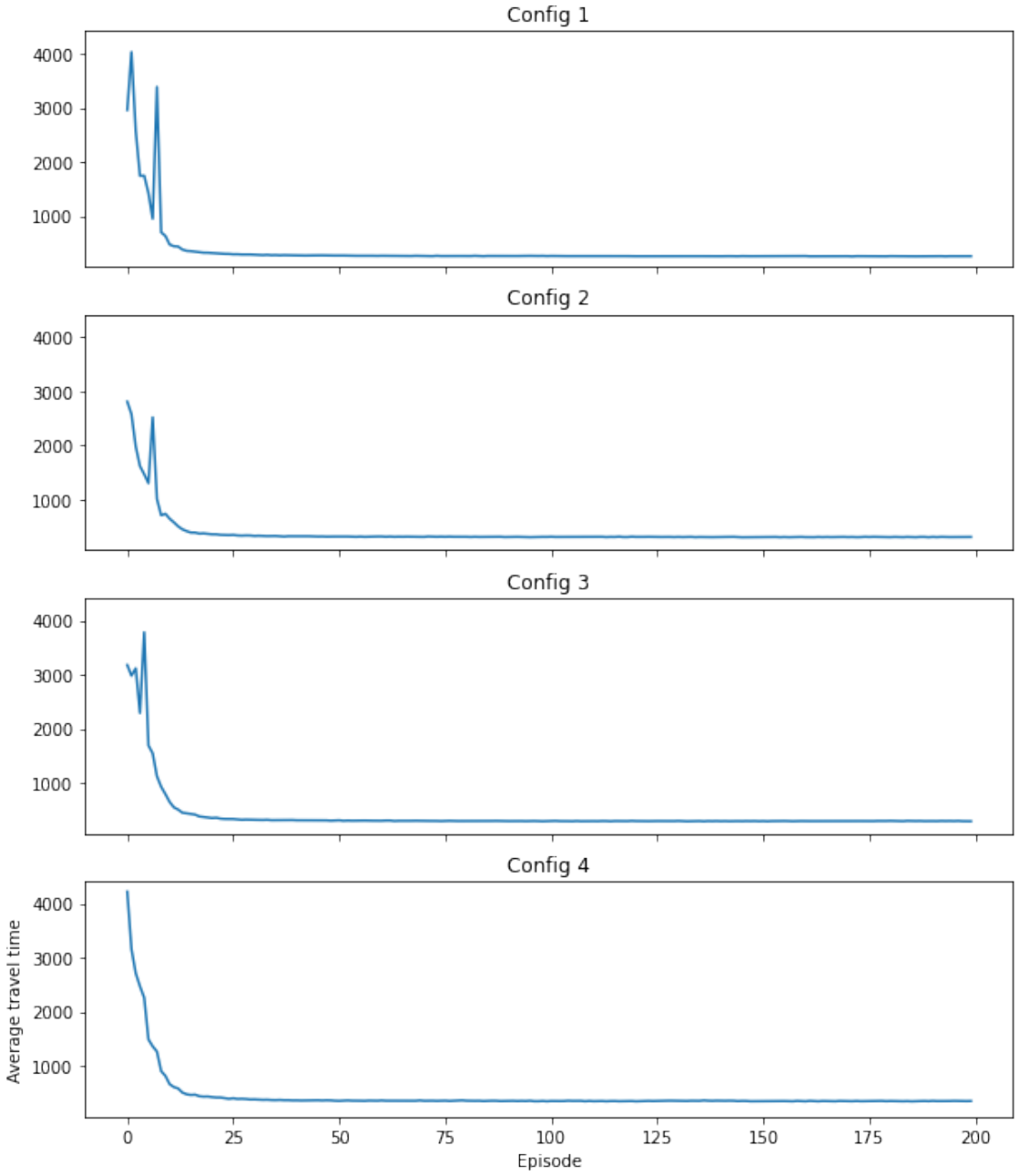

Figure 4.1: Average travel time during each learning episode for all configurations

We can see that the algorithm converges after about 50 episodes of training but upon closer inspection of the values depicted we can see that performance continues to improve slightly even after episode 50.

4.2 Effects of using Deep Implicit Coordination Graphs and Modified GAE

The following results compare the performance the approach using the DICG architecture as the critic with vanilla GAE and the approach that uses DICG architecture and the proposed modified GAE. Training was performed for 50 episodes using config 4 of the dataset. The following figure compares per episode the average travel time for each approach, the figure has a log scale on the y axis.

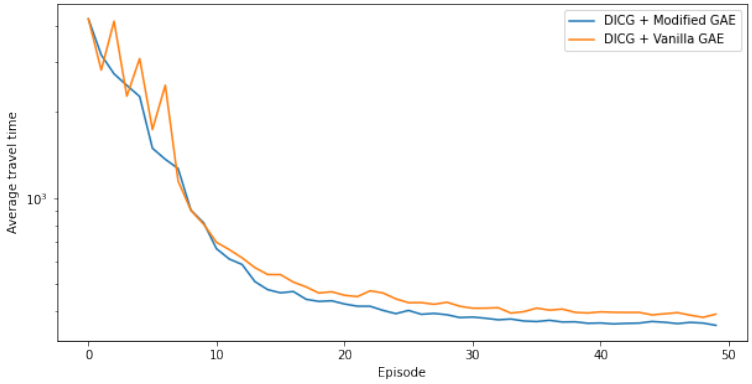

Figure 4.2: Average travel time for GAE+DICG vs Modified GAE+DICG. Episodes 0-50

We can see in the figure that during the first 50 episodes the proposed modified GAE is more stable as we can see some fluctuations in the vanilla approach, it also has faster convergence rate and it is clear from the figure that by episode 50 the modified approach has reached a much better performance compared to the vanilla approach. Additionally, after the depicted episodes, the modified approach continues to improve at a better rate than the vanilla GAE, this becomes more apparent by looking at the following figure which shows the performance during episodes 50-100 of training.

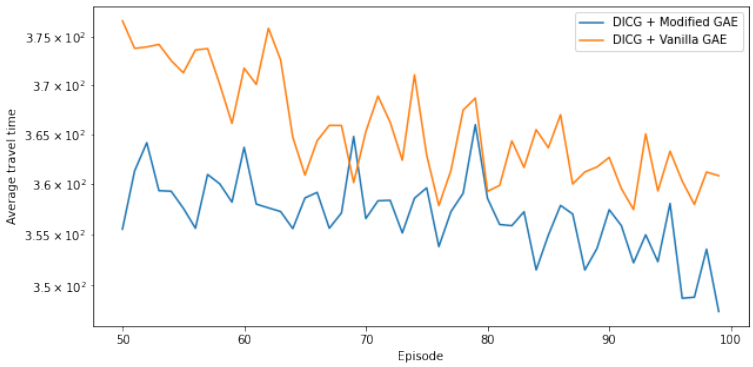

Figure 4.3: Average travel time for GAE+DICG vs Modified GAE+DICG. Episodes 50-100

It is worth noting that when training the agent using DICG and vanilla GAE for 200 episodes it achieves 354.9 average travel time which is worse compared to the 347.6 average travel time possible using the modified GAE for advantage estimation, further demonstrating the improvement in performance attributed to this modification.

The following figures compare using the DICG architecture with vanilla GAE to using a basic MLP as the critic. Figure4.4 shows the average travel time of the first 50 episodes of training.

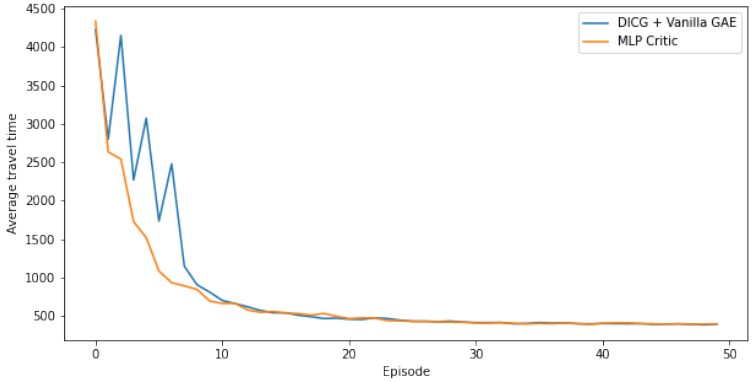

Figure 4.4: Average travel time for GAE+DICG vs MLP critic. Episodes 0-50

The plots show similar performance and even better convergence properties for the MLP critic, that is due to the simplicity of the model. However, upon closer inspection of the training process for the remaining episodes it is clear that the DICG approach continues to improve performance whereas the MLP critic plateaus. The following figure that shows the results of episodes 50-100 of training.

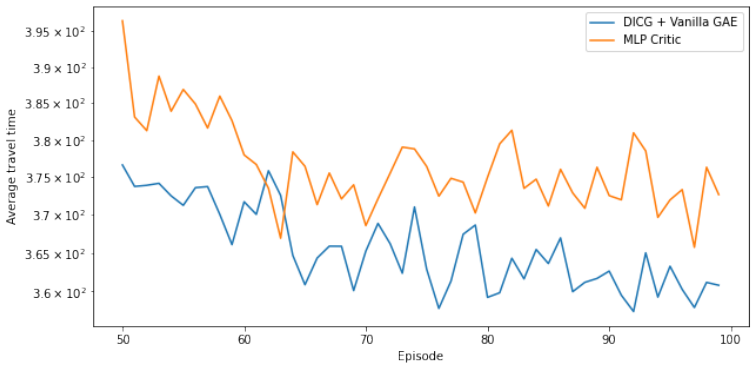

Figure 4.5: Average travel time for GAE+DICG vs MLP critic. Episodes 50-100

The same trend continues until episode 200. At the end of training the MLP critic achieves 361 average travel time during greedy execution which is worse in comparison to the 354.9 achieved by the DICG architecture.

The logs containing the metrics over 200 episodes for all approaches are available in the GitHub repository linked in section 3.10.

Chapter 5

Conclusion

This thesis applies the Deep Implicit Coordination Graphs architecture to the problem of Traffic Signal Control, furthermore it proposes a modification to Generalized Advantage Estimation that better fits the task and attempts to improve credit assignment.

The results show the improvement to convergence rate and to the overall performance when using the proposed advantage estimation when compared to the vanilla GAE approach.

5.1 Advantages

Aside from performance one of the main advantages of the applied approach is its actor-critic nature. After training the critic is no longer needed and the actor can be deployed to operate the signals. In this work most of the model complexity is in the critic whereas the actor is a simple multilayer perceptron that is shared between all intersections, i.e. the same network is trained using the data of all agents, this reduces the complexity and improves sample efficiency. In addition, the approach uses a decentralized policy, meaning each agent selects an action based on its own observations, the decision making is not centralized and thus the policy network can be deployed to each agent independently.

5.2 Limitations

The applied approach is an on-policy algorithm and naturally it shares the limitations of this family of algorithms, these algorithms can only learn from the data generated by applying the current policy, hence they can not learn from old data and can not utilize techniques such as experience replay [8]. As a result these methods struggle with sample efficiency however they are more stable than off-policy algorithms.

Another limitation is that while the architecture and algorithm scale well for large numbers of agents, the current implementation of the approach has the need to loop over all agents to calculate the value of $\hat{s}_t^{\neg a_t^i}$ which adds to the time required to perform a training iteration.

5.3 Future Work

Because this approach was proposed with multi-agent large road networks in mind it displayed acceptable performance with some improvement over other methods when evaluated using a 16 intersection network, it is expected that the approach would potentially have even better performance when applied to even larger road networks consisting of dozens or hundreds of intersection, this work does not apply the approach to such a road network.

This thesis applies the centralized-training-decentralized-execution architecture of DICG which has its own advantages discussed in the previous section, another possibility would be to use a centralized policy and use observations of all agents to take actions which might improve performance but it would lose some of the advantages of the decentralized policy.

Since this work focused on the architecture and algorithm and not on the state representation and reward function design, a clear future direction would be to utilize a more sophisticated definition of the RL task, for example chen et al. [15] report that simply using the concept of pressure in the design of the reward and state they were able to improve the performance of several different algorithms. In addition, a thorough hyperparameter search for this approach was not performed due to time constraints, a proper search can potentially achieve even better results, and this

includes applying a lower action interval Δt and tuning the hyperparameters for that interval.

Another aspect that can potentially boost performance is using Recurrent Neural Networks as part of the actor and the critic architecture to allow the agent to exploit any temporal dependencies, for example in our approach the agent observing the number of vehicles waiting in each lane is unaware of how long each vehicle has been waiting, this piece of information directly affects our objective to minimize the average travel time and could be useful to the agent for making decisions. Despite the advantage these types of architectures can provide, they can be especially challenging to train and tune [21].

When it comes to the modified advantage estimation, there are quite a few ways to improve the proposed approach. The applied approach calculated advantages off of the value of an afterstate having the action of the agent being evaluated removed by replacing the phase in the state representation with its previous value. An approach worth investigating would be to calculate advantages off of the expectation of the values over the afterstates taking into account all possible actions and their probabilities.

Appendix A

Hyperparameters

A.1 Synthesized 4-by-4 dataset hyperparameters

The following hyperparameters were used in all of the experiments applied to this dataset.

- $\gamma = 0.75$
- GAE $\lambda = 0.95$
- $\Delta t = 20$
- Critic learning rate= 0.0001
- Critic training epochs = 4
- Critic training mini batch size = 20
- Actor learning rate= 0.0001
- Actor training epochs= 2
- Actor training mini batch size = 128
- $c_2 = 0.001$
- DICG embedding size = 16
- DICG GCN number of layers is 3

The layers and neurons of the DICG encoder are described in table A.1.

Layer	Type	Neurons	Activation
1	Hidden	124	ReLU
2	Hidden	64	ReLU
3	Hidden	32	ReLU
4	Output	16	Tanh

Table A.1: Layer details of the DICG encoder

The layers and neurons of the DICG MLP are described in table A.2.

Layer	Type	Neurons	Activation
1	Hidden	512	Leaky ReLU
2	Hidden	256	Leaky ReLU
3	Hidden	128	Leaky ReLU
4	Hidden	64	Leaky ReLU
5	Output	1	None

Table A.2: Layer details of the DICG MLP

The layers and neurons of the actor MLP are described in table A.3.

Layer	Type	Neurons	Activation
1	Hidden	512	Tanh
2	Hidden	256	Tanh
3	Hidden	128	Tanh
4	Hidden	64	Tanh
5	Output	Number of phases	None

Table A.3: Layer details of the actor MLP

For the details please see the implementation referred in section 3.10

Bibliography

[1] INRIX. *INRIX 2020 Global Traffic Scorecard*. 2020. URL: https://inrix.com/scorecard/.

[2] Hua Wei et al. "PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. KDD '19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 12901298. ISBN: 9781450362016. DOI: 10.1145/3292500.3330949. URL: https://doi.org/10.1145/3292500.3330949.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[4] Huichu Zhang et al. "CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario". In: (May 13, 2019). DOI: 10.1145/3308558.3314139. arXiv: 1905.05217v1 [cs.MA].

[5] Tommi Jaakkola, Satinder P Singh, and Michael I Jordan. "Reinforcement learning algorithm for partially observable Markov decision problems". In: *Advances in neural information processing systems* (1995), pp. 345–352.

[6] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[7] Gerald Tesauro. "TD-Gammon: A Self-Teaching Backgammon Program". In: *Applications of Neural Networks*. Ed. by Alan F. Murray. Boston, MA: Springer US, 1995, pp. 267–285. ISBN: 978-1-4757-2379-3. DOI: 10.1007/978-1-4757-2379-3_11. URL: https://doi.org/10.1007/978-1-4757-2379-3_11.

[8] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: (Dec. 19, 2013). arXiv: 1312.5602v1 [cs.LG].

[9] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[10] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: (July 20, 2017). arXiv: 1707.06347v2 [cs.LG].

[11] John Schulman et al. “High-Dimensional Continuous Control Using Generalized Advantage Estimation”. In: (June 8, 2015). arXiv: 1506.02438v6 [cs.LG].

[12] Sheng Li et al. “Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning”. In: (June 19, 2020). arXiv: 2006.11438v2 [cs.LG].

[13] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: (Sept. 9, 2016). arXiv: 1609.02907v4 [cs.LG].

[14] Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: (Jan. 25, 2016). arXiv: 1601.06733v7 [cs.CL].

[15] Chacha Chen et al. “Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (2020), pp. 3414–3421. DOI: 10.1609/aaai.v34i04.5744.

[16] Hua Wei et al. “A Survey on Traffic Signal Control Methods”. In: (Apr. 17, 2019). arXiv: 1904.08117v3 [cs.LG].

[17] Fabio Pardo et al. “Time Limits in Reinforcement Learning”. In: *PMLR 80: 4042-4051 (2018)* (Dec. 1, 2017). arXiv: 1712.00378v3 [cs.LG].

[18] Greg Brockman et al. “OpenAI Gym”. In: (June 5, 2016). arXiv: 1606.01540v1 [cs.LG].

[19] Huichu Zhang, Markos Kafouros, and Chang Liu. *tlc-baselines*. https://github.com/zhc134/tlc-baselines. 2020.

[20] Hua Wei et al. “CoLight: Learning Network-level Cooperation for Traffic Signal Control”. In: (May 11, 2019). DOI: 10.1145/3357384.3357902. arXiv: 1905.05717v2 [cs.MA].

[21] Bram Bakker. “Reinforcement Learning with Long Short-Term Memory”. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. NIPS’01. Vancouver, British Columbia, Canada: MIT Press, 2001, pp. 14751482.

List of Figures

4.1 Average travel time during each learning episode for all configurations 23
4.2 Average travel time for GAE+DICG vs Modified GAE+DICG. Episodes 0-50 24
4.3 Average travel time for GAE+DICG vs Modified GAE+DICG. Episodes 50-100 25
4.4 Average travel time for GAE+DICG vs MLP critic. Episodes 0-50 . . 25
4.5 Average travel time for GAE+DICG vs MLP critic. Episodes 50-100 . 26

List of Tables

3.1 Table listing the technologies used. 18

4.1 Table containing results over the synthesized 4by4 dataset. 21

A.1 Layer details of the DICG encoder 31
A.2 Layer details of the DICG MLP 31
A.3 Layer details of the actor MLP 31