CLD771 Minor Project Report

Evaluation of B.Tech. Minor Project

Name: Samarth Bhatia Entry Number: 2019CH10124

Project Title: Using Deep Reinforcement Learning for scheduling gasoline blending and distribution (SGBD)

Supervisor: Prof. Hariprasad Kodamana

Report 1: 17 September, 2021

Summary of work done:

My supervisor, Prof. Kodamana told me to go through 2 papers (linked below) and to solve the problem presented in Paper 1 using DRL methods, as done in Paper 2.

My work so for was to

- thoroughly read the 2 papers Paper 1 Paper 2.
- Paper 1, "Scheduling of gasoline blending and distribution using graphical genetic algorithm", was about the problem of SGBD using a GGA. They consider certain variables and parameters associated with SGBD such as flow rates of products, costs of components, demand of orders, costs of changeovers of products in blenders and product tanks, and tardiness costs etc., which makes for a close-to-reality situation.
- Paper 2, "A deep reinforcement learning approach for chemical production scheduling", applies Deep Reinforcement Learning to a production problem in Chemical Engineering in the form of a **A2C (Advantage Actor Critic) algorithm**, which involves 2 neural networks, one for the actor (which decides what to do based on the current state), and one for the critic (which approximates a value function to objectively tell how good the actions taken by the actor are).
  This has recently been quite popular in DRL because the **Actor-Critic** model eliminates the disadvantages of using **Action-Value** methods (e.g. **Q-Networks** and **Deep QN**) like *not being performant for continuous spaces*, and also of using **Policy-Gradient** methods (e.g. the **REINFORCE** algorithm) like *only being able to update the policy after a whole episode of learning* since it is basically a Monte Carlo simulation (No Temporal Difference Learning).
- I also learned a lot about Reinforcement Learning and Deep Reinforcement Learning from various lectures and an online course (Udacity DRL). Some of what I have learnt includes:
  1. interfacing with the OpenAI `gym` python package which is universally used in RL problems and also as a benchmark as it contains many standardized environments.
  2. The Bellman equation, a universal equation in RL applications:

$$\underbrace{\text{New}Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underset{\text{New Q-Value}}{\alpha}\Big[\underbrace{R(s,a)}_{\text{Reward}} + \underset{\text{Discount rate}}{\gamma}\ \overbrace{\max_{a'} Q'(s',a')}^{\substack{\text{Maximum predicted reward, given}\\ \text{new state and all possible actions}}} - Q(s,a)\Big]$$

  3. **Action-Value** method Q-Networks (and DQNs) and my implementations of them (GitHub) on the MountainCar-v0 standard `gym` environment
  4. **Policy-Gradient** method REINFORCE algorithm and my implementation of them (GitHub) on the CartPole-v0 standard `gym` environment

- In addition to this, I have started work on the project part - I have started to implement the environment used in Paper 1, which will take some time to complete and will need to be changed as required (if the training implementation requires something differently) or because of optimizations in general.
  Note that this is the main part of the implementation describing the problem statement and every single constraint for proper results, and the training/agent part would not be very different from normal. However, I will have to change the agent slightly too to be compliant with the environment.

Future Plan:

- In the immediate future, I will try and work out all the kinks of completing the environment. I also want this environment to be compatible with the `gym` package so that I would be able to release an open-source version of this SGBD problem (which will of course have less detail) as a python package for others to use.
- Once the bulk of the environment is done, I will head to establishing some baseline results, considering the performance of the Graphical Genetic Algorithms in Paper 1, and also using prebuilt implementations like the `stable-baselines` package, the `catalyst`package (specialized for DRL using PyTorch for the NN part), etc.
- After establishing baseline scores I will try different algorithms such as **A2C (Advantage Actor-Critic)**, **A3C (Asynchronous Advantage Actor-Critic)**, **PPO (Proximal Policy Optimization)** (which is a **Policy-Gradient** method) and others accordingly. I don't see much scoe for Deep Q-Networks for SGBD since most of the variables and actions are in the continuous space.