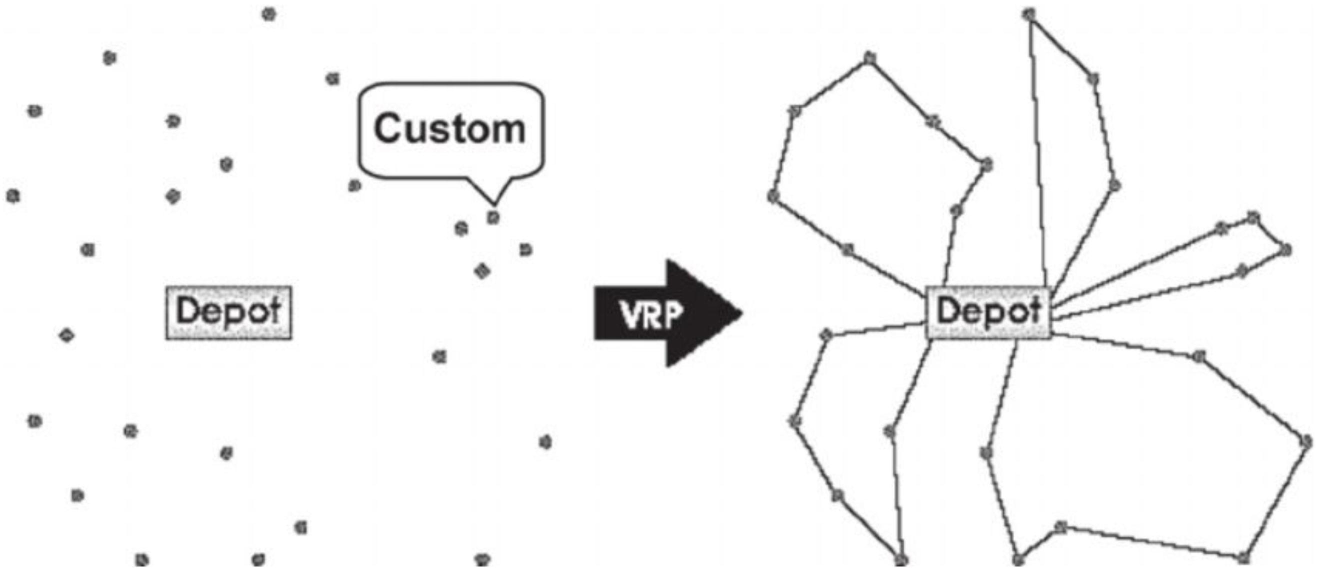
School Bus Routing Problem using CVRP

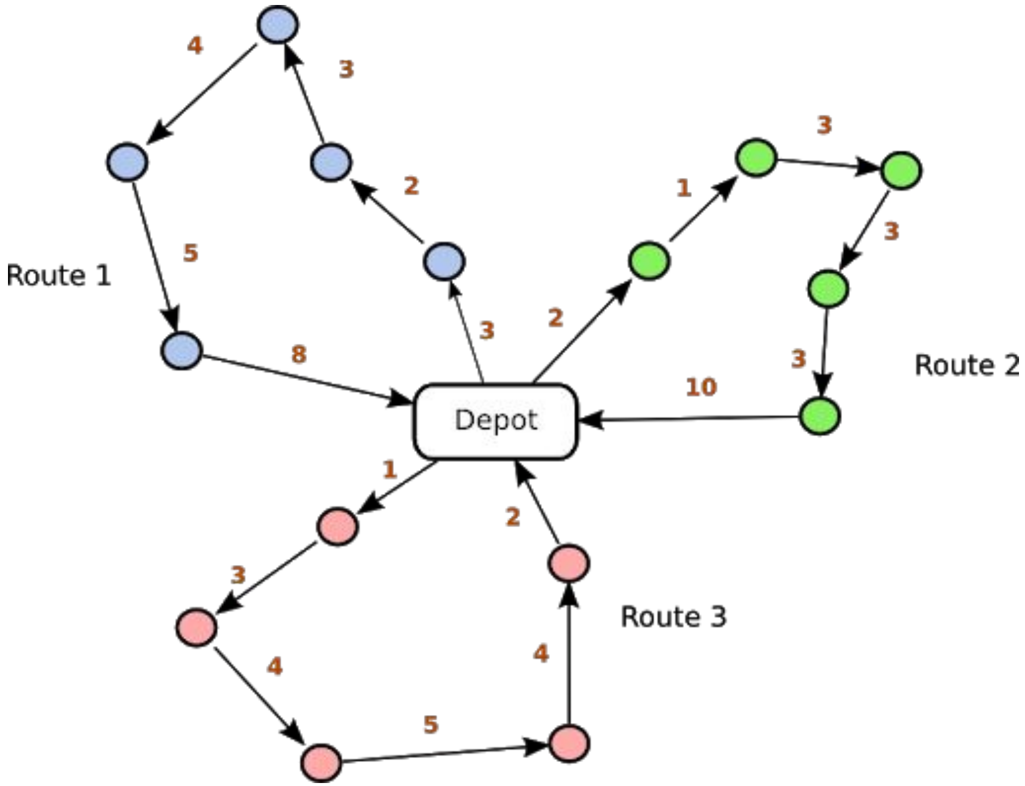
Niranj Jyothish (njyothi)
Nitya Sankarakumar (nsankar3)
Poorvaja Penmetsa (ppenmet)

Vehicle Routing Problem (VRP)



Capacitated VRP

Vehicle(s) with capacity go back to depot to unload current load and reset capacity.



Problem into Consideration - School Bus Routing

In school bus routing problems only potential stops are given, and selection of stops and determining bus routes depends on students' locations and capacity of each bus.

Depot → **School**
Demand → **Number of students at potential stops**

Goal is to determine:

1. what set of stops to visit
2. for each student which stop he or she must walk to
3. routes that lie along the chosen stops, so that the total travelled distance is minimized.

Dataset

Under Instances -> instance files

Name	Status	Date modified	Type	Size
my1		1/22/2018 7:18 PM	Text Document	1 KB
my2		1/22/2018 7:18 PM	Text Document	1 KB
sbr1		1/22/2018 7:18 PM	Text Document	9 KB
sbr2		1/22/2018 7:18 PM	Text Document	9 KB
sbr3		1/22/2018 7:18 PM	Text Document	16 KB
sbr4		1/22/2018 7:18 PM	Text Document	16 KB
sbr5		1/22/2018 7:18 PM	Text Document	16 KB
sbr6		1/22/2018 7:18 PM	Text Document	16 KB
sbr7		1/22/2018 7:18 PM	Text Document	16 KB
sbr8		1/22/2018 7:18 PM	Text Document	16 KB
sbr9		1/22/2018 7:18 PM	Text Document	16 KB
sbr10		1/22/2018 7:18 PM	Text Document	16 KB

Smaller Instances

Format of data shown:

Stop / Student ID	X - coordinate	Y - coordinate

Content of instance files:

```
6 stops, 25 students, 20.000 maximum walk, 25 capacity

0 50.000 50.000
1 38.390 30.261
2 21.710 34.625
3 22.467 21.108
4 38.726 79.167
5 33.491 66.206


1 26.080 36.624
2 42.858 22.531
3 36.392 32.102
...
(shortened)
...
24 52.590 62.867
25 30.624 58.687
```

0 denotes school

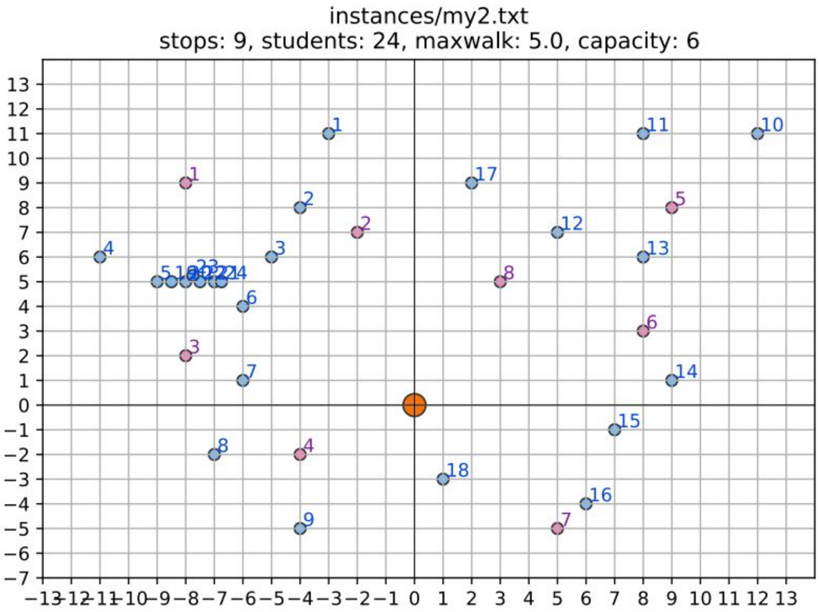
6 bus stops coordinates

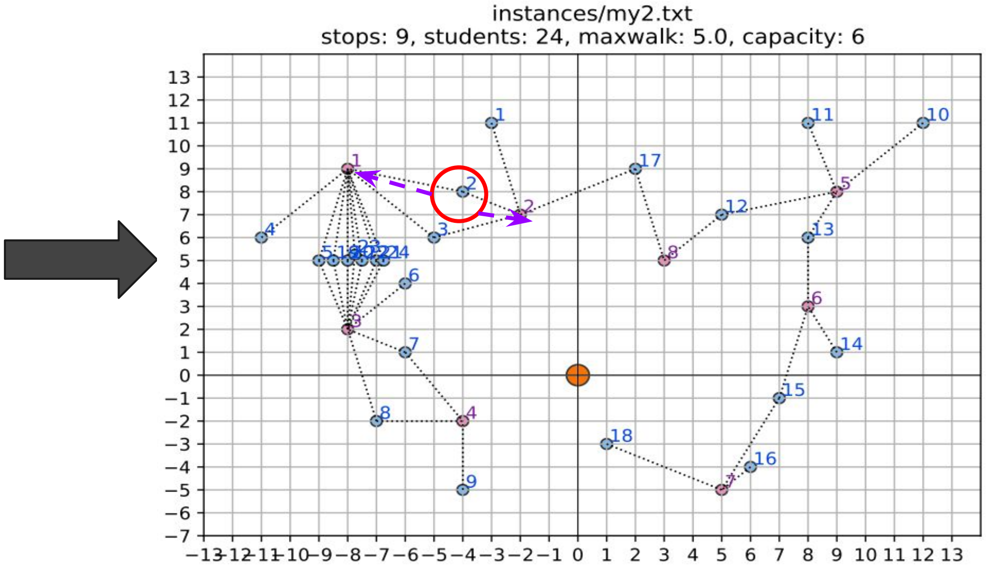
25 student coordinates

Constraints and Assumptions

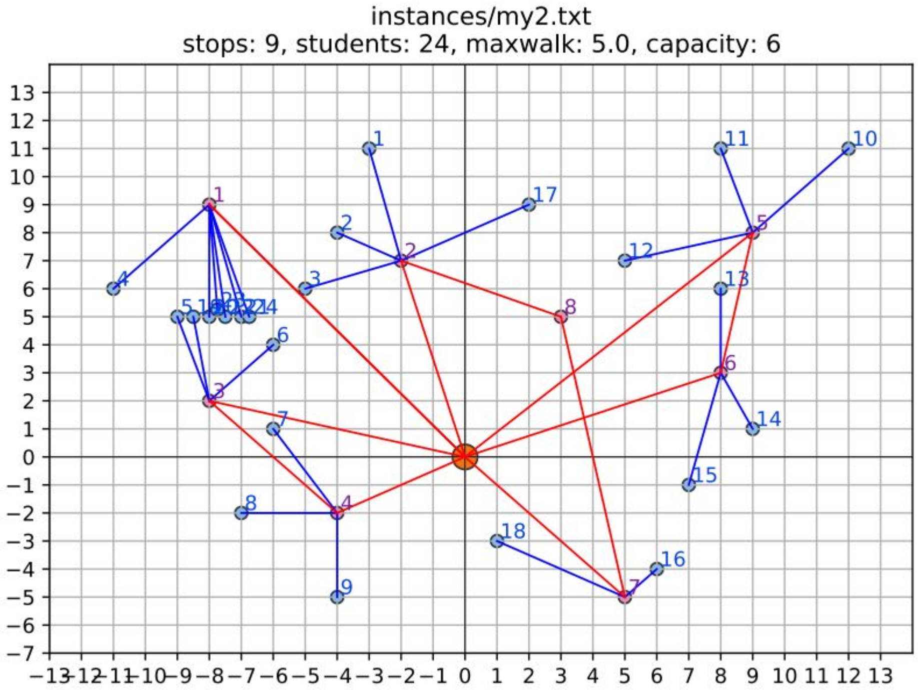
1. Student access stop only within Maximum walk
2. Fixed Capacity for Bus
3. A stop is visited by only ONE bus.

Problem into Consideration - School Bus Routing

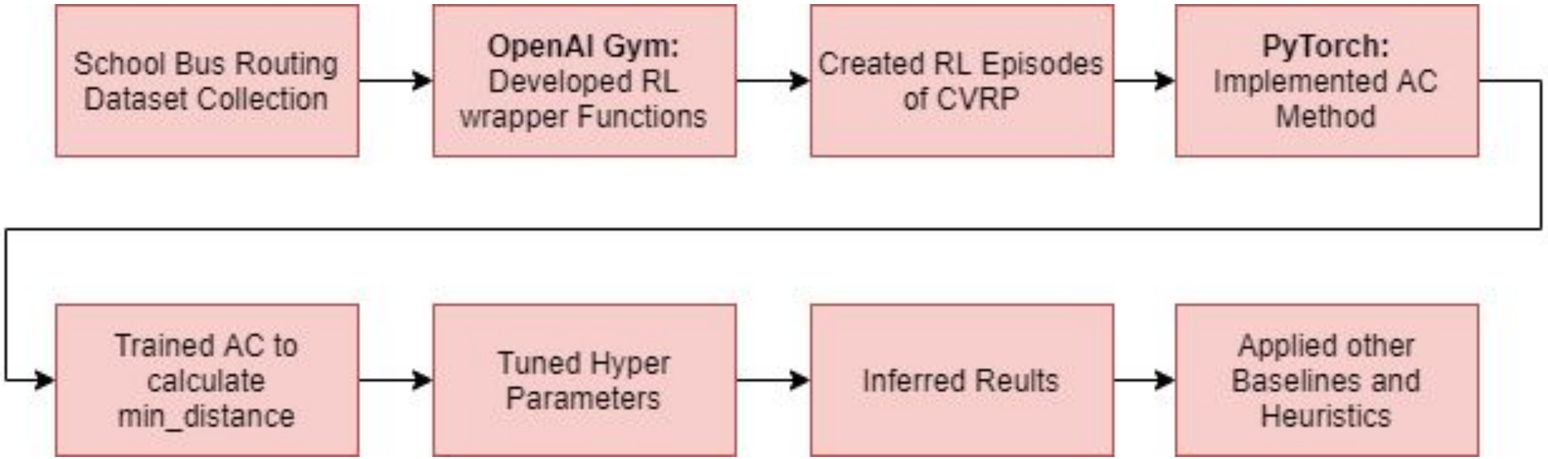




Routes

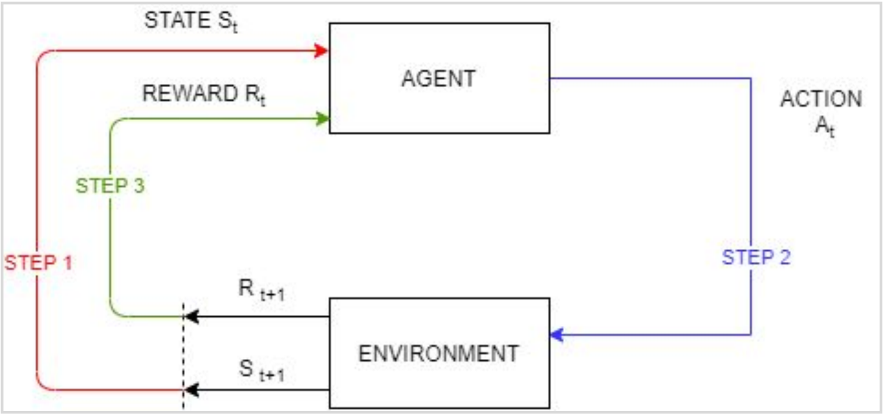


Project Flow Diagram

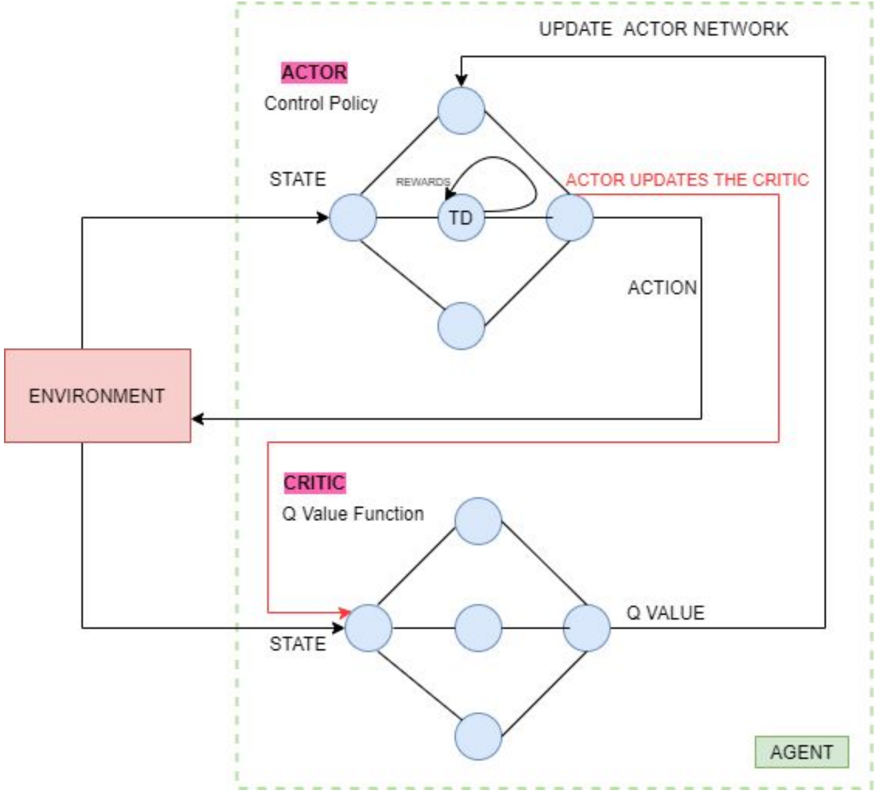


Concepts Involved

Reinforcement Learning Mechanism



Actor Critic Network



Main Parameters of Open AI Gym Environment

01	**Action Space**	• There are 3 discrete actions: ○ Action 2: Depot to Node ○ Action 0: Node to Node ○ Action 1: Node to Depot
02	**Observation Space**	• State: Demand, capacity, current point, visited • Info: stopId, stop coordinates, maxwalk
03	**Demand**	• Number of students at each bus stop
04	**Reward**	• Negative of route length till the point. • Huge Penalty for non-ideal actions
05	**Done and Episode**	• Done: demands have become zero (we have picked up all the students) • Episode: Set of steps taken to achieve done

OpenAI GYM Environment

Init and Reset:

All the initial parameters for the environment are set in the init constructor. once an episode terminates, the reset function resets all the parameters defined in the init. A sample of our observation space parameter:

INITIAL OBSERVATION

	stop_id	stop_x	stop_y	maxwalk	demand	capacity	visited	current_point
0	0	0.0	0.0	5.0	0	6	0	8
1	1	-8.0	9.0	5.0	3	6	0	8
2	2	-2.0	7.0	5.0	4	6	0	8
3	3	-8.0	2.0	5.0	6	6	0	8
4	4	-4.0	-2.0	5.0	3	6	0	8
5	5	9.0	8.0	5.0	3	6	0	8
6	6	8.0	3.0	5.0	1	6	0	8
7	7	5.0	-5.0	5.0	3	6	0	8
8	8	3.0	5.0	5.0	1	6	0	8

Action 0

Node to Node

Can only occur when **current point is not at depot(0)** and reward is weighted negatively worse as capacity decreases teaching the system to not pursue this action when remaining capacity is 0.

Observations updated:

- Current point = new node
- Demand at current node = 0 OR
  Demand = demand - capacity
- Capacity = capacity - demand OR
  capacity = 0

Action 1

Node to Depot

Occurs when **current point is not at depot(0)** and reward is weighted negatively worse as capacity increases teaching system to not pursue this action when capacity is not 0.

Observations updated:

- Current point = 0 (stop id)
- Demand at current node = 0 OR
  Demand = demand - capacity
- Capacity = full (all unloaded at depot)

Action 2

Depot to Node

Occurs only if the **curent_point is depot(0)** since it should start from the depot and the bus is empty or at full capacity to pick students.

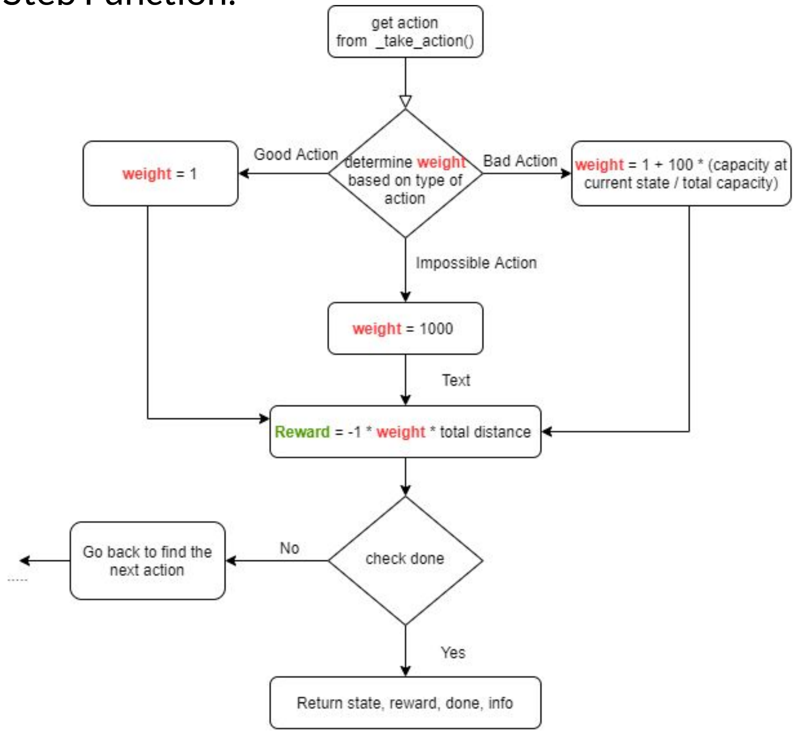
Observations updated:

- Current point = new node
- No demand at depot
- Capacity = capacity - demand OR
  capacity = 0

Open AI Environment

```python
class CustomEnv(gym.Env):
  """Custom Environment that follows gym interface"""
  metadata = {'render.modes': ['human']}

  def __init__(self, arg1, arg2, ...):
    super(CustomEnv, self).__init__()

    # Define action and observation space
    # They must be gym.spaces objects

    # Example when using discrete actions:
    self.action_space = spaces.Discrete(N_DISCRETE_ACTIONS)

    # Example for using image as input:
    self.observation_space = spaces.Box(low=0, high=255, shape=
                    (HEIGHT, WIDTH, N_CHANNELS), dtype=np.uint8)

  def step(self, action):
    # Execute one time step within the environment
    ...

  def reset(self):
    # Reset the state of the environment to an initial state
    ...
```

Step Function:



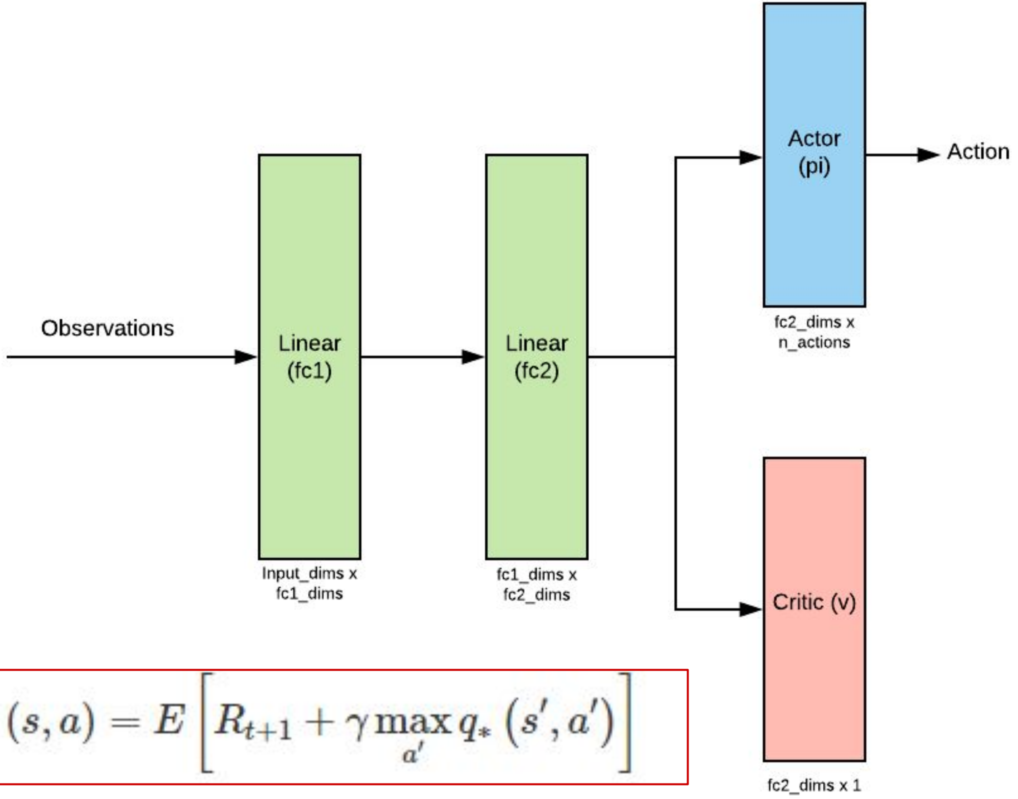
Actor Critic Algorithm

Choose action by ACTOR

- get policy from actor_critic.forward(observation)
- get policy from **softmax**(policy)
- get action probabilities from **T.distributions.Categorical**(policy)
- sample the action probabilities
- get **log_probs** from action_probs.log_prob(action)

Learn by CRITIC

- get critical_value of next state from actor_critic.forward(state_)
- calculate **reward**
- calculate **deltaQ** : q*(s,a) - q*(s', a')
- calculate actor loss as **-self.log_probs** * **delta**
- calculate critic loss as **delta**2**
- Backward prop on critic and actor loss

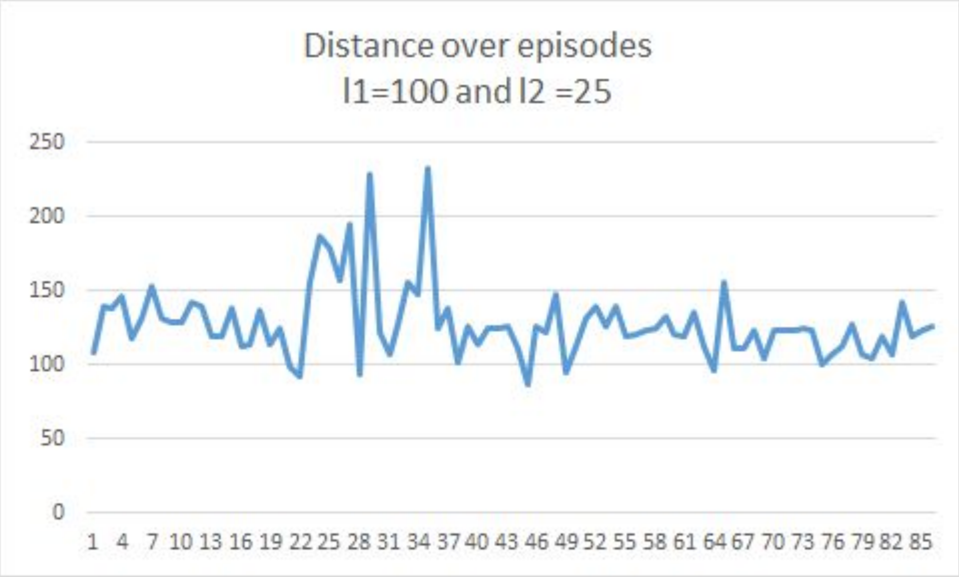


$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

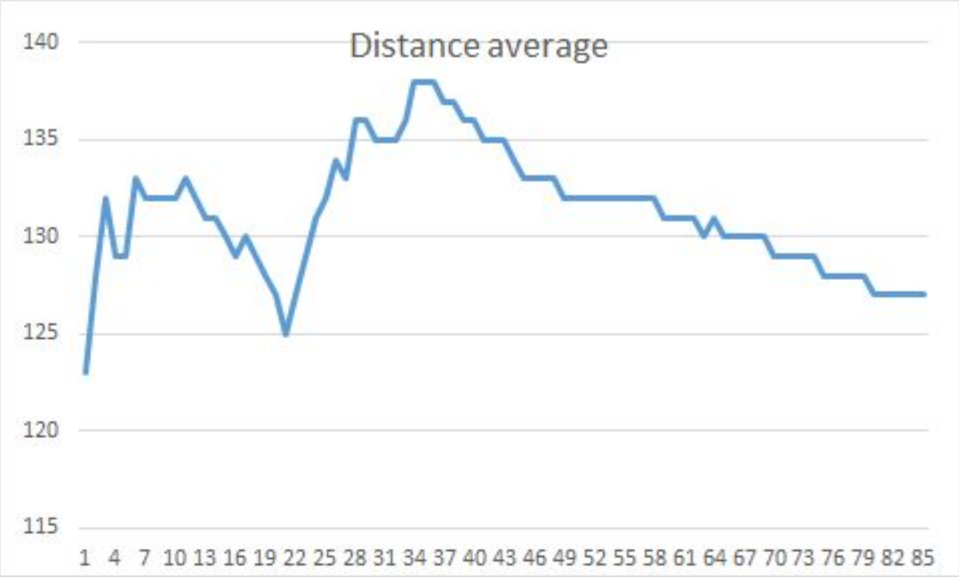
Hyper Parameters

Parameter	Value	Description
learning_rate	0.00001	Learning Rate: Proportional to how fast the agent can learn. It denotes how quickly the agent can abandon the previous Q value for the new Q value.
Gamma (discount rate)	0.95	Discount Rate: Influences the agent to account for more immediate rewards
Layer1_size (fc1_dims)	50	Fully connected layer 1 size
Layer2_size (fc2_dims)	10	Fully connected layer 2 size

Results



Distance over episodes
l1=100 and l2 =25
250
200
150
100
50
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85



Distance average
140
135
130
125
120
115
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85

Baselines and Heuristics

	Random Restart with Greedy	Clarke-Wright Savings	Google-OR	Actor Critic
Instance	My2.txt (9 stops, 24 students, 5.0 maxwalk, 6 capacity)	My2.txt (9 stops, 24 students, 5.0 maxwalk, 6 capacity)	My2.txt (9 stops, 24 students, 5.0 maxwalk, 6 capacity)	My2.txt (9 stops, 24 students, 5.0 maxwalk, 6 capacity)
Number of Iterations	100	1	1	200
Shortest path	[[6, 5], [4, 3], [2, 8, 7], [1]]	[0, 1, 5, 0, 0, 2, 6, 0, 0, 3, 2, 0, 0, 4, 1, 0, 0, 5, 7, 0, 0, 6, 3]	0 Load(0) -> 4 Load(3) -> 3 Load(6) -> 0 Load(6)	[1, 0, 0, 4, 0, 6, 8, 5, 0, 7, 2, 3]
Minimum Distance calculated	112.17076974694193	106.35776267107448	17 (single local path distance)	81.667744
Time taken to run	0.04688 seconds	1.5113 seconds	2.117 seconds	5 minutes (total)
Verdict	Good but not optimal. Since it follows a random restart, no path exists sometimes.	Good, but not optimal since it uses only uses pairs of nodes even when capacity is not full.	Not a good heuristic since this always considers a fleet of vehicles with different capacities and not just one.	Good, but more complex than other algorithms.

Summary and Future Work

1. Extensive research done on various way to solve VRP using Reinforcement Learning.
2. Understood Capacitated VRP and other variants of VRP
3. Analyzed implementation of School Bus Routing Problem with CVRP
4. Learnt about basics of RL,Q- learning and Policy gradient methods such as Actor Critic
5. Created OpenAI Environment specific to the bus routing problem
6. Basics of Pytorch and CUDA DNN
7. Developed Actor Critic method in PyTorch
8. Inferred results and tuned hyper parameters to give optimal results
9. Applied Baseline and Heuristics such as Google OR, Random Restart with greedy and Clarke-Wright Savings on a fixed data

Future work

1. Node to depot when capacity != 0 good action when depot is next closest node
2. Compare with Deep Q-learning

Questions?

References

1. Policy Gradients in a Nutshell
2. Reinforcement Learning: Bellman Equation and Optimality (Part 2)
3. Understanding Actor Critic Methods and A2C
4. Intuitive RL: Intro to Advantage-Actor-Critic (A2C)
5. Deep Reinforcement Learning for Routing a Heterogeneous Fleet of Vehicles
6. Create custom gym environments from scratch — A stock market example
7. OpenAI Gym
8. Reinforcement Learning w/ Keras + OpenAI: Actor-Critic Models
9. Landing on the Moon with Discrete Actor Critic Methods (Pytorch Tutorial): https://www.youtube.com/watch?v=53y49DBxz8U
10. Deep Lizard - Reinforcement Learning - Goal Oriented Intelligence: https://deeplizard.com/learn/playlist/PLZbbT5o_s2xoWNVdDudn51XM8lOuZ_Njv
11. Solver for Capacitance Vehicle Routing Problem - School bus routing problem with bus stop selection
12. OptMLGroup/VRP-RL: Reinforcement Learning for Solving the Vehicle Routing Problem
13. VehicleRoutingProblem/cvrpImprovedImpl.py at master · shlok57/VehicleRoutingProblem