15XD81– REINFORCEMENT LEARNING

Title	**Airline Scheduling**
Link to the data	https://openflights.org/data.html
Domain	**Vehicle Routing problem**
Roll Number & Name	**16PD05 – Aswath Rao & 16PD28 – Ridhanya**
Tools used	**Google Colab for scripts**
Source code	https://github.com/ridhanya99/Reinforcement-Learning

Project Overview

In this project, we will try to implement an algorithm that will schedule airlines such that it maximise the satisfaction of passengers using basic reinforcement learning agent which is being trained using off policy TD approach (Q learning). Essentially Q learning lets the agent use the environment's rewards to learn, over time the best action to take in a given state. In this approach, we train a sing policy model that find near optimal solutions for a board range of problem instances, only by observing the reward signals and following feasibility rules. The trained model produces solution as the number of passengers who are satisfied by the agents scheduling. This project is an implementation of variation of Vehicle Routing problem (VRP) since it has the potential to be applied more generally to combinatorial optimization problems.

In the following, we describe our application domain and the tools used in this project.

Data

To train RL agent, the program relies on two types of data: simulated and actual data.

1.) **Simulated data-** It is a simple airport data with few sample cities having randomly generated distances, passengers with stimulated requirement of moving from one city to another and airplanes with random capacity allocated.

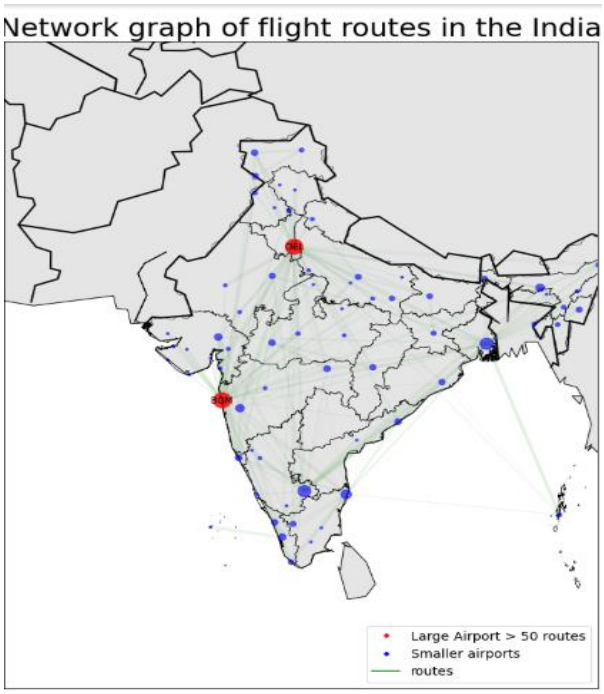
2.) **Actual data-** OpenFlights - Airport, Plane, and Route Databases to Schedule flights for user. The Airports database contains over airport 10,000 airports spanning the globe; the Route Database contains routes between airports on airlines. Unfortunately, the route database is not very up-to-date. It currently contains 59,036 routes between 3,209 airports on 531 airlines spanning the globe. and the plane database contains a curated selection of 173 passenger aircraft with IATA codes covering the vast majority of flights operated today commonly used in flight schedules and reservation systems. From this data we have only taken Indian country data for simplicity. The program also generated passengers requirement data based on real airport and routes data.

To find distance between airports we have used "scipy.spatial.distance" library in python.

Data link https://openflights.org/data.html

Data Visualization

In this section, we will look at the flight route network between airports in India. The goal is to represent vertices (airports) and edges (flight route) and preserve the geographical relationships between different vertices. I have used two python library **basemap** and **networkx** for this visualization. Here we also differentiate the airports with their number of incoming and outgoing flights, by making airports with lots of flight will have larger size and appear more visible on the map.



Tools

Google Colab - Used to write python scripts.

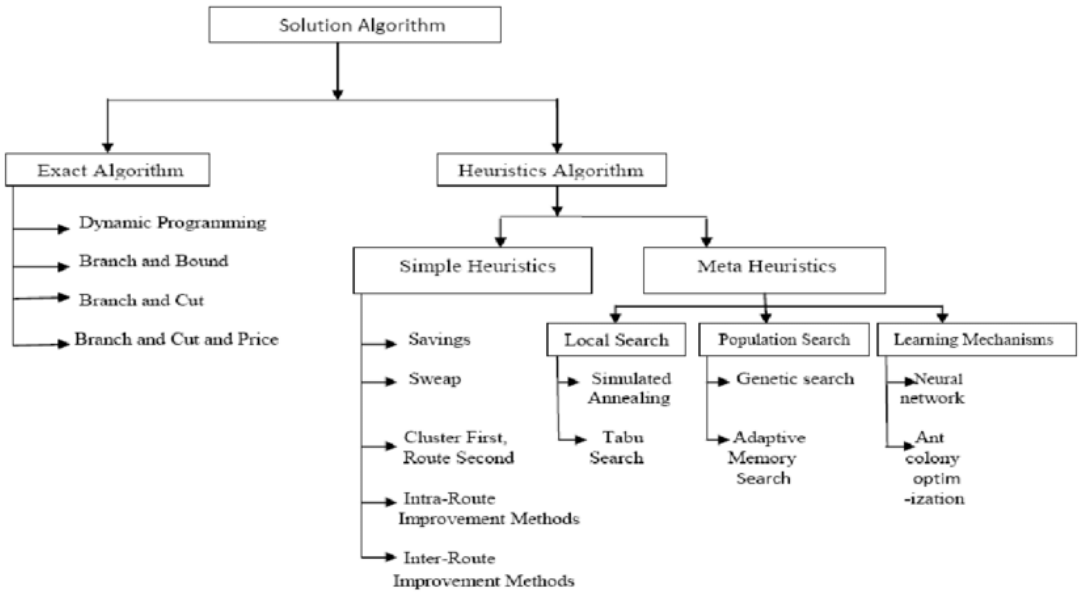
Basemap- A python library to visualizing airport data and routes data with the context of map.

Alternative Tools

- Integer Programming – "A Modelling Language for Mathematical Programming" (AMPL) a high level programming language that translates mathematical statement that describes a mathematical program into a format readable by most optimization software packages.
- PDDL – Planning Domain Definition Language (PDDL) is an attempt to standardize artificial intelligence planning languages. **pddlpy,** A python library that provides a very simple interface with domain problems.

Background and Related Work

We are not the first people to model scheduling of airplane, however to our knowledge this approach of using Reinforcement learning is novel. As this problem is a variation of vehicle routing problem (VRP), there are various tools such as Google-OR tools and technique like,



In this project we are giving RL agent control over scheduling on environmental information including data about airport, routes and planes. This is where our work will be novel and hopefully, show results indicating that RL can be used to increase profit for airplane organisation.

Terminology

- **Agent -** Reinforcement learning object acting as a "Airplane scheduler"
    - **Policy-** agent's behaviour function, which is a map from state to action.
    - **Value Function-** a prediction of future rewards.
    - **Model-** agent's representation of the environment.
- **Environment-** An airplane object that will provide feedback such as the number of satisfied passenger and reward or penalty.
- **State-**The number of happy passenger at a given time.
- **Training**- Interactions between the agent and environment for the agent to learn what the goal is and how to achieve it the best.
- **Episode**- Number of independent training session (the environment is reset, but agent keeps the learning from one episode to another)
- **Session**- Each session has multiple episodes with both environment and agent reset; the goal is to benchmark agent performances based on the number of episodes (e.g. will more training episode leads to high success ratio? When should we stop the training?).
- **Q-Table**- A matrix the agent use to decide future action based on state-action-reward tuples; the agent develops this Q-Table from each training episode based on environment feedback.

Problem Formulation

We begin the following section with a description of an application specific problem formulation at a theoretical level. We describe how we formulate our problem within the context of a general reinforcement learning problem, which includes a state space, action space, reward function and goal state. We proceed this formulation with a description of our implementation plan.

Theoretical Problem Formulation

State Space- We define state space S as a set of states S1,S2,.....Sn $\in$ S as a collection of values corresponding to the following: {cities, people, airplanes, city distance }. (i.e. The planes and passengers in the city)

Action Space- We define our action space as which passengers assigned to each plane.

Reward Function- Reward and penalty is structured as the following:

- +2000 → if all passengers are assigned flights.
- +50 → if the action assign vehicle to city where there are passengers.
- +100*x → if action satisfied x passengers (x = no of satisfied passenger by taking an action)
- -50 → if action has not satisfied any passengers.
- -cost → here cost is the cost for flight (i.e. distance between cities).

The reward structure encourages the agent to keep the scheduling within an acceptable range while taking actions.

Goals- The overall goal of the agent is to maximize the long term reward. This reward is a weighted sum of all future expected rewards starting at the current state.

Solution Method

This section provides a detailed overview of solution method used in our project.

Q-Learning

To learn the mapping between action and weighted future reward at a given state, Q-learning is the most basic and traditional learning method. This technique does not require a model of the environment. Q-learning can handle problems with stochastic transitions and rewards, without requiring adaptions. For any finite Markov decision process (FMDP), Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable. Q-learning can identify an optimal action-selection policy for any given FMDP. The following equation, inspired by the Bellman's equation is used to develop a mapping between state, action, and expected future value.

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Where,

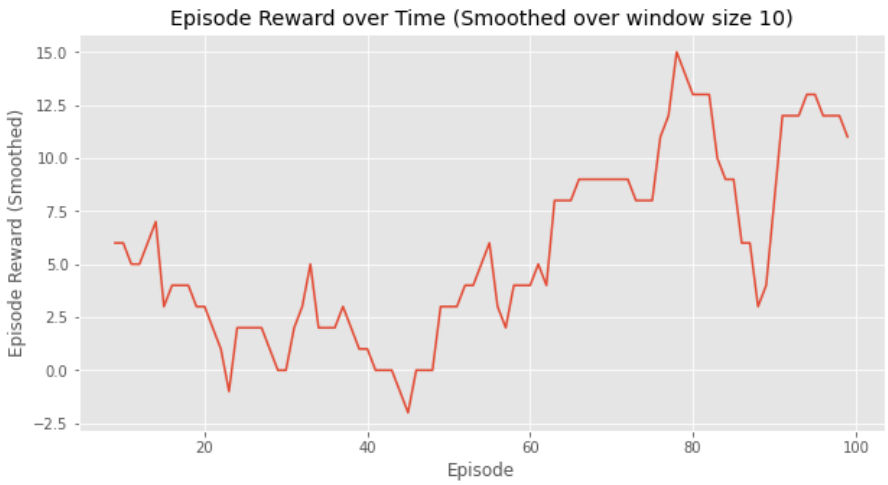
State = s, Action = a, Reward = r, Time step = t

Key Performance Indicator

We measure if the computer agent can achieve the following without deliberate human programming, but only with reward and penalty based on scheduling.

1. **Low Cost:** Total cost the agent used to schedule flights.
2. **Short Learning Time:** The agent should be able to develop new rebalancing strategy quickly when external objective changes (e.g. no of airports, no of passengers, airplanes in city etc.)
3. **High Success Ratio:** Percentage of time the agent satisfies the requirement of maximum passengers.

Result



Evaluation Techniques

We will evaluate the lower level performance of our system by looking at multiple factors, some of which are still to be determined. Ultimately the most important factors will be profit, passenger satisfaction, time for scheduling etc.

Conclusion

The team has proven the preliminary application of RL approach to Airline Scheduling. The RL agent was able to develop, adopt and improve airline scheduling autonomously in various country data. By evaluation metrics the Q-Learning methods achieve better performance.

Ultimately, from Airline perspective, the best choice of model depends on their priorities. What if passengers are willing to go to any intermediate airports before reaching their desired destination? Then the Q-learning methods may warrant additional consideration.

References

- https://papers.nips.cc/paper/8190-reinforcement-learning-for-solving-the-vehicle-routing-problem.pdf
- https://www.academia.edu/33997000/The_Vehicle_Routing_Problem_An_overview_of_exact_and_approximate_alg