Approximating the Chinese Postman Problem using Reinforcement Learning

Austin Byrd

University of Colorado Colorado Springs

Abstract

The application of reinforcement learning to real world problem statements has become increasingly prevalent in the field of artificial intelligence. As analytical methods break down on larger scale systems the solutions grow combinatorially. Recent breakthroughs in combinatorial optimization problems such as Vehicle routing problems (VRP) (Nazari et al., 2018), give promising applications to reinforcement learning approximations of optimal solutions to problems that require non-polynomial time algorithms to solve or approximate. The application has been applied to the Traveling Salesman Problem (TSP), which is a nodal routing problem, while less attention has been paid to edges and arc routing problems, such as the Chinese Postman Problem (CPP) until recently. Recent advancements have been made on the CPP with extension of Load dependent costs CPP-LD. The purpose of this paper is to use reinforcement learning techniques such as the markov decision process (MDP) to solve an arc routing problem using the traditional CPP as the instance of an arc routing problem. In other words, to create an agent that can be put into a CPP valid graph and can learn on the graph to approximate an optimal policy to solve the CPP for that graph.

Introduction

The Chinese Postman Problem was first posed by a chinese mathematician named Kwan Mei-Ko in 1962 (Sokmen et. al, 2019). It encapsulates the issue of a Postman who must deliver the area's mail. To do this the Postman must start at the postoffice, go down every street, and after having gone down every street return to the postoffice.

This problem has many real world applications such as: circuit testing, robotic path planning, inspections, distributed ledgers, etc.

The problem can be stated more formally as such:

$$G = (V, E)$$

G is a undirected weighted graph made of the set V of vertices, and the set E of weighted edges.

The objective of the CPP is to find the **shortest or lowest weight** path that

1. Starts and ends at the same node ($V_0 = V_f$)
2. Crosses every edge in the graph (Path crosses all $e \in E$)
3. Minimizes the cost of the weighted edges taken in the walk (circuit).

The problem takes place in a graph, and the agent must begin at the start node and cross edges until all edges have been crossed and the agent has returned to the start node. The agent may cross edges multiple times in its path, but to complete a solution it must return to the start node.

The problem focuses upon Non-Eulerian trail containing graphs as it can cross edges multiple times. This is an important difference betwen the CPP and TSP.

The statement of the problem includes the optimality condition of **shortest or lowest weight** path. Which a reinforcement learning algorithm can approximate.

Analytical methods have approximately solved the problem and heuristic methods have improved upon the solution space (Pan et al., 2024). The continuing issue with those methods is that as the problem graph grows the algorithm runtime grows exponentially. This poses the question: can a reinforcement learning algorithm match the approximality of analytical methods when applied to a traditional Chinese Postman Problem? And, if so, can it scale to larger problem graphs?

Background

Reinforcement learning is a type of machine learning that relies upon the agent or machine to learn from past actions and rounds of actions to create a policy that can guide its future actions. This allows for an agent to be immersed into an environment and make actions. From those actions values can be calculated for each action taken from a given state in the environment. This ability to learn from the environment by being immersed in the environment and making actions allows for knowledge to be collected and applied when the same choices are made again.

These tools allow for the formation of a policy that can outperform analytical methods on certain problems.

Algorithm 1: Chinese Postman Problem Algorithm

Input: $G = V, E$ where G is a Non-Eulerian Graph
Output: Eulerian Graph

1: Find the degree for each node in the graph.
2: Pair odd degree nodes together greedily minmizing path distance.
3: Add edge between each pair of odd degree nodes. Making every node degree even.
4: Find Eulerian Circuit for the graph.
5: Sum the length of the Eulerian circuit.
6: Apply Hierholzer's Algorithm
7: Return Eulerian Circuit length.

The application of reinforcement learning to arc routing problems could yield algorithms that operate on policies that human ingenuity could not make but could only be made by large amounts of experience.

Related Work

The Chinese Postman Problem is a NP-Hard problem that has been attempted with analytical approximations and heuristically. Due to the NP-Hard nature of the question the solutions grow exponetially with the number of odd degree edges in the graph. See Algorithm 1. for the traditional algorithm.

The time complexity of the method outlined in Algorithm 1. is set by the basic operation that is step 3 of the algorithm as the number of possible pairs of odd degree nodes grows combinatorially. This has been approximated via greedy methods but this does not provide exact solutions.

Real world application of the Chinese Postman Problem include Load dependent costs which are that the load to be delivered gradually diminishes as the edges are crossed improving the cost of edges that are traversed later in the delivery cycle. This can have resource savings in real applications of fuel usage and travel time.

These real world applications have inspired many methods to solving the CPP and many more complicated problem spin offs of the CPP. These more complicated methods have been formulated for more complicated versions of the CPP such as CPP-LD, Rural Postman Problem, Directed CPP, Time Dependent CPP, Multiple CPP, etc. (Sokmen et al., 2019) (Christofides et al., 1981) (Pan et al., 2024) (Corberan et al., 2018) (Corberan et al., 2021) (Tran et al., 2023) (Terefe, 2015). These problems have many different methods that are used to approximate optimal solutions. A variety of methods are included in a few notable libraries [1] (Corberan et al., 2021).

The huge majority of these methods have been analytical, and recently heuristic and meta-heuristic methods have been showing great promise including reinforcement learning.

The application of a model known as Arc-DRL has been the most effective (Tran et al., 2023), Arc-DRL is an Arc based Graph traversal algorithm that uses Graph-Attention based Deep Reinforcement Learning to train a Graph-Attention Encoder that produces embeddings to pass through some number (N) of 2 layer attention layers and then to the decoder. The decoder is intended to reconstruct the optimal tour of the input graph. This model has done exceptionally well at solving the CPP-LD. The neural network was trained using 2 datasets (Corberan et al., 2018).

Additional models that have been used on this problem include: the Genetic algorithm, Ant Colony algorithm, Greedy Heuristic Construction, Iterated Local Search, and Variable Neighborhood Search as well as others.

Many of these models utilize cutting edge technologies and models in the field of Machine Learning and some a multi-model hybrid algorithm that have many layers of neural network (Vaswani et al., 2017).

IV. Methodology

The type of reinforcement learning algorithm proposedo for implementation is a Q-Learning model to learn on a specific graph. This method is implemented by transforming the CPP into an MDP. To transform the Chinese Postman Problem into a Markov Decision Process there must be a clear set of States, Actions, Rewards, a Return function, a discount factor, and a policy. Python will be used for program development.

These will map to the Chinese Postman Problem via:

States

Set of states S: Nodal Position of Agent in the graph $v \in V$
Goal State s_f: When all other nodes are reached and the agent has returned to the initial state. $s_0 = s_f$

Actions

Set of actions A: The agent can move to any Edge connected to current Node Position $e \in E_s$

Rewards

Set of rewards $R = \{100, 0\}$ Agent receives a reward of zero until it reaches the goal state where it receives a reward of one hundred.

Discount Factor

A discount factor of $\gamma = .8$ will be applied, a low discount factor is used to highly encourage the agent to immediately accomplish the goal of solving the CPP.

Return Function

Return Function $V_\pi(s)$, $Q_\pi(s, a)$ etc. (Value, Goodness, or Quality Expectation of a state or given action in a state)

The return function is trained on the episodes of training the agent goes through. The function is trained on the averaged value of a state in an episode by utilizing the discount

[1] (https://github.com/Olibear/ArcRoutingLibrary and https://www.uv.es/corberan/instancias.htm)

factor to reduce the reward of a state if it takes many actions to reach the goal state rather than it taking fewer actions to reach the goal state via a different action/state.

This numerically ranks the states in a vectorized mapping between states and values this is the value function $V_\pi(s)$.

$Q_\pi(s, a)$ is a matrix of values mapping state action pairs to an expected return value.

Policy

Policy π: Initially equally distributed random actions are selected at any given state. This initial random policy is used to collect the data needed to calculate $V_\pi(s)$, $Q_\pi(s, a)$. As many more episodes are ran and the averages of the values in $V_\pi(s)$, $Q_\pi(s, a)$ stabilize the policy can be stabilized.

Methodological Philosophy

The methodological outlook is to formulate a simple model. To test its capabilities by hand. Then to simulate the MDP. This will train the return functions on episodes of training from a single graph. This initially training of the model on one small and simple graph that has been solved by hand ensures that the model converges to the expected solution.

Once the algorithm is successful upon small examples of the CPP, the model will be applied to create more policies for more complex graphs with larger number of nodes and edges for the algorithm to run upon. This creates a different model for every graph.

The final goal will then be to attempt to create a more general algorithm using Deep Q learning that will create a Deep set of Q values that will be trained on many different graphs creating a neural network that takes in a graph and outputs the approximately optimal solution. This is done with the goal of the model to be able to solve CPP examples without having trained upon the graph. This is to model a tourist trying to walk down every street but not knowing the area very well as opposed to the postman that comes to know the area well.

The model will be trained off of solutions generated from random graphs and solved with the traditional pairing method outlined in Algorithm 1.

The goal of the general algorithm is the supersede the traditional analytical algorithm in runtime on new input graphs to solve.

Evaluation

The evaluation of the reinforcement learning algorithm will be done by comparing the solutions that the Return function converges to, to a solution calculated with the traditional analytical approximation methods for solving CPP.
The evalution method utilized is outlined below.

- The agent will traverse all graph edges and return to the starting node.
- The number of edges traversed will be calculated at the end of each episode and the reward of 100 is given for reaching the end.
- The discount factor rewards the agent for completing the episode quickly and thus incentivices the policy to be efficient.
- Thus the value the agent receives at the end of the episode is due to how many edges it had to traverse to complete the episode.
- As the π converges to approximately optimal we will compare it to Algorithm 1. to find the accuracy.
- The goal is for the reinforcement learning algorithm converged solutions performance ratio to be $\rho \leq 1.1$ or within 10% of the approximations from Algorithm 1.
- Additionally the Deep-Q learning RL algorithms performance will be measured by both the the ratio of the approximation of minimum path length but also by the runtime of the RL and analytical algorithms. This combined runtime and approximation will balance speed and optimality goals of the agent.

Timeline

- Complete creation of Q-learning algorithm on single graph 2/25
- CompleteTest and Evaluate Model 3/3
- 3/18 Midterm Presentation
- 3/31 Complete creation of Deep Q learning algorithm dataset for training
- 4/14 Complete training of Deep Q learning Model
- 4/28 Complete Evaluation of Deep Q Model
- 5/1 Final Presentation

Conclusions

This proposal has been focused upon implementing a reinforcement learning agent that will traverse CPP graphs and learn from its attempts. Training a return function $Q_\pi(s, a)$. That will guide the transformation of our policy until it has converged to values that are approximately optimal for the given CPP graph. This initial prototype will aid in the formulation of a more general algorithm that will yield a more general model to dynamically solve unbeforeseen CPP examples. The final model will generalize on graphs of a limited size.

References

Tran, Cong Dao, and Truong Son Hy. 2023. Graph Attention-based Deep Reinforcement Learning for solving the Chinese Postman Problem with Load-dependent costs. arXiv preprint arXiv:2310.15516

Sokmen, Ozlem and Emec, Seyma and Yilmaz, Mustafa and Akkaya, Gokay. 2019. *An Overview of Chinese Postman Problem.*, International Conference of Advanced Engineering Technology

Pan, P., Zhu, H. 2024. *Approximation algorithms for the restricted k-Chinese postman problems with penalties.* Optim Lett 18, 307–318 . https://doi.org/10.1007/s11590-023-01992-z

Farahani, Reza Zanjirani, 2012. *Graph Theory for Operations Research* and Management: Applications in Industrial Engineering: Applications in Industrial Engineering. IGI Global.

Christofides, N., Campos, V., Corberán, A., and Mota, E. (1981). *An algorithm for the rural postman problem.* Report, International Conference for Operations Research.

Corberán, Á., Eglese, R., Hasle, G., Plana, I., and Sanchis, J. M., 2021. *Arc routing problems: A review of the past, present, and future.* Networks an International Journal.

Corberan, A., Erdoğan, G., Laporte, G., Plana, I., and San- chis, J. (2018). *The chinese postman problem with load- dependent costs.* Transportation Science, 52(2):370–385

Terefe, Abel, 2015. *Arc routing in household waste collection*, Aalto University.

Acknowledgments

We would like to thank Tom Lehrer for all of his support. Thank you for reading my paper.