Multi Objective Q-Learning for the Chinese Postman Problem

Austin Byrd

Abstract

The application of reinforcement learning to real world problem statements has become increasingly prevalent in artificial intelligence. As analytical methods break down on large scale systems, the solution space grows combinatorially. Recent breakthroughs in combinatorial optimization problems, such as Vehicle routing problems (VRP) (Nazari et al., 2018), give promising applications to reinforcement learning to problems that as of now 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 edge and arc routing problems, such as the Chinese Postman Problem (CPP) until recently. Recent advancements have been made on CPP with extension of Load dependent costs CPP-LD by applying Graph-Attention based Deep Reinforcement Learning. The purpose of this paper is to use reinforcement learning to create an agent that can be embedded into a CPP valid graph and can learn on the graph to approximate an optimal policy to solve the CPP for that graph by using a Multi-Objective Q-learning framework.

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:

$$G = (V, E)$$

Where 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. Begins and Terminates 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 problems main focus is upon Non-Eulerian 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.

Analytical methods have approximately solved the problem and heuristic methods have improved upon the most optimal attainable solution space (Pan et al., 2024). The continuing shortfalling of the methods is that as the problem graph grows the algorithm runtime grows exponentially. This poses the question, can a reinforcement learning algorithm match the approximal optimality of analytical methods when applied to a traditional Chinese Postman Problem? And, if so, can it be altered to generalize to any CPP valid problem graph?

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.

The application of reinforcement learning to arc routing problems could yield policies that human ingenuity could not make, but only could 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 (Sokmen, et. al. 2019). Due to the NP-Hard nature of the question the time complexity of algorithmic solutions grow exponetially with the number of odd degree edges in the graph. See Algorithm 1. for the traditional algorithm.

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 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 (CPP-LDC). CPP-LDC relies upon that the load to be delivered gradually diminishes as the edges are crossed, lowering 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 of solving the CPP and many more complicated problem spin offs of the CPP. Expansive methods have been formulated for more complicated versions of the CPP such as CPP-LD, Rural Postman Problem, Windy CPP, 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 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 from a related experiment (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).

The application of Multi-Objective optimizations has grown in popularity for problems such as the CPP. The Pareto Front is a subset of the solutions come to by an algorithm that are most optimal for balancing multiple objectives (Van Moffaert, Nowe 2014). This most optimal subset of solutions is saved and only superceded when solutions of higher optimality for both objectives are observed. Then the Pareto Front is updated to the most optimal solution and the algorithm continues searching.

The CPP being a two stage problem requires a consideration of two objectives. The first being to cross all the edges of the graph, and the second being to return to the start node. Therefore there are two objectives that must be accomplished. This multi-objective problem requires Multi-Objective optimization applications such as Multi Objective Markov Decision Process (MOMDP) and Multi Objective Q-learning (MOQ) to solve (Yang, et. al 2019) (Hayes, et. al 2021). MOMDP and MOQ have been formulated and applied to many problems in Reinforcement Learning, but until now MOQ has rarely been applied to the CPP (Majumder 2019).

Methodology

The type of reinforcement learning algorithm proposed for implementation is a Multi-Objective Q-Learning model to learn on a specific graph. This method is implemented by transforming the CPP into an MOMDP. To transform the Chinese Postman Problem into a Multi-Objective Markov Decision Process there must be a clear set of States, Actions, Vectorized Rewards, a Scalarized Return function, a discount factor, a learning rate, and a policy (Van Moffaert, Nowe 2014). Python was used for program development.

Multi Objective Reinforcement Learning

MOQ learning is different from traditional Q-learning in many ways. The most important is the reward vectorization, the Q value vectorization the results in a policy formulated from the scalarization of the Q vector, and the alteration of the action selection method.

These will be applied to the Chinese Postman Problem via the following components.

Objectives

The agent has two objectives the first objective Ξ is to cross all edges in the graph.

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

The second Υ is to return to the start state.

Therefore the Set of Objectives O for the CPP is defined as such

$$O = \{\Xi, \Upsilon\}$$

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 across any Edge connected to current Node Position $e \in E_s$

Actions in MOQ are made via a Transition function T. T is a probability distribution function among the possible actions in a state, where a state, action pair is input and the output is the probability of that action being taken from that state following the Q values $Q^O_\pi(s, a)$ over all objectives.

Rewards

Set of rewards $R = \{-1, 0, -e_i\}$ Agent receives a two rewards, one for each objective $o \in O$.

Where $-e_i$ is the weight or cost of crossing a given edge.

For objective Ξ the agent recieves a reward of 0 for reaching the start/goal state, and a reward of -1 for all other actions.

For objective Υ the agent recieves a reward of $-e_i$ for crossing edges. The agent recieves a reward of 100 for crossing all the edges in the graph.

Discount Factor

A discount factor of $\gamma = .9$ will be applied, a low discount factor is used to highly encourage the agent to immediately accomplish the goal of solving the CPP.

Learning Rate

A learning rate of $\alpha = .9$ is applied and gradually decreased until it is at .1 to start learning quickly and rapidly slow the learning.

Return Function

Return Function $Q^O_\pi(s, a)$ etc. (For a given Objective there are distinct functions for the Value, Goodness, or Quality Expectation of a state or given action in a state)

For MOQ with two objectives there are two Q tables one for each objectives Q values. Both Q tables are kept in one larger vectorized Q table. Each objectives Q tables are trained on that objectives return. This updates the objectives Q tables simultaneously on a single state, action pair but updates both Q tables on that action depending on the return for that action given the objective.

The return functions are trained on the episodes of training the agent goes through. The functions are trained on the averaged value of a state in an episode by utilizing the discount 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 combined with small negative rewards highly incentivize the agent to end the episode quickly for highest reward. This numerically ranks the states in a vectorized mapping between states and values for given objectives.

$Q^O_\pi(s, a)$ is a matrix of values mapping state action pairs to an expected return value **for an objective**.

Transition Functions

The transition function $(T(s, a, s'))$ is a function that takes in the current state s, the selected action a, and the possible next states s'. The function outputs a probability distribution of what s' will be after the action is enacted. This is most interesting for MOQ when applied to non-deterministic environments. This is due to the probabilistic outcomes of certain actions. This paper strictly entails deterministic transition functions and therefore it is not required for action selection preference as s' is predestined by (s, a).

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 $Q^O_\pi(s, a)$ which is then updated as the quality of state action pairs are calculated for both objectives.

The combination of these objective Q tables into one final policy can be done in multiple ways. The chosen method is via Pareto Dominance, a method of identifying the most optimal solutions of the Pareto Front.

The final policy is derived by taking either the maximum action between both Q tables, or to take the average of of the Q tables, and then follow the maximum of the resultant Q table.

For this experiment the latter method was utilized.

Methodological Philosophy

The methodological outlook is to formulate a MOQ-learning RL model. Test the models capabilities on a curated CPP with a known optimal solution. Then to simulate the MDP. This will train the return function on episodes of training from a state space; meanwhile collecting a Pareto Front. This initial training of the model on one graph that has a known solution ensures the evaluation of the model's capabilities before being applied to larger CPP instances.

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

Evaluation

The evaluation of the reinforcement learning algorithm will be carried out by comparing the solutions of the trained final policy; that was formulated from Q values informed by the Pareto Front, to a solution calculated with the traditional analytical approximation methods for solving CPP.

The evalution method utilized is a comparison made between Single Policy MORL, and the traditional analytical algorithm.

- The benchmark of Single Policy MORL methods will be compared to the results quantified by finding the performance ratio between analytically solution methods and each of the MORL methods.
- The goal is for the MORL algorithms policies performance ratio to be $\rho \leq 1.1$ or within 10% of the approximations from Algorithm 1 respectively.

.

Graphs

Graph 1 is a curated small graph of 5 nodes and 8 edges

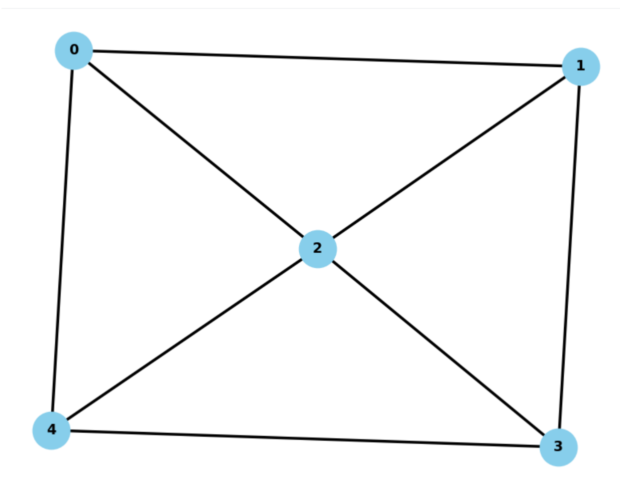

Figure 1: Small Graph

Graph 2 is a randomly generated medium sized graph of 25 nodes and 40 edges

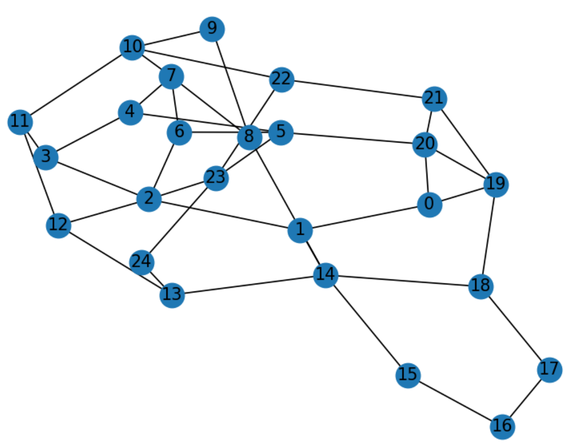

Figure 2: Medium Graph

Graph 3 is a randomly generated large graph of 100 nodes and 150 edges

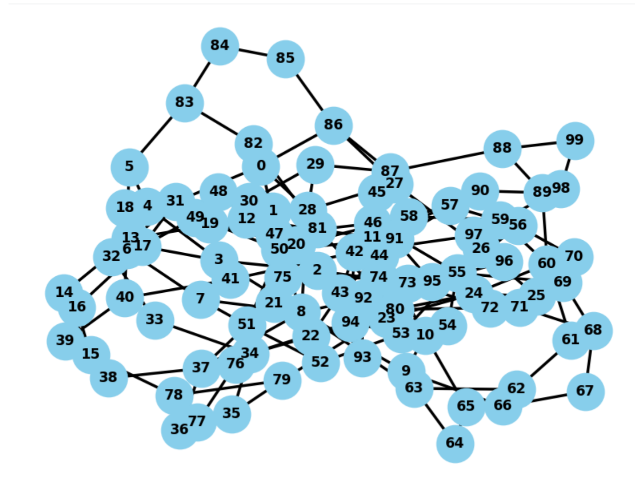

Figure 3: Large Graph

Results

The evaluation of the model is done with the above three graphs, Figures 1, 2, and 3. Figure 1 is the small graph, Figure 2 is the medium graph, and Figure 3 is the large graph. The solution space that is observed in the final policies and Pareto front of single policy MORL agents solving the CPP are approximately equivalent to traditional analytical methods. With the caveat that larger graphs have a larger temporal cost to solving and yield less optimal solutions than analytical methods. The Pareto Frontier is a subset of solutions that are the maximally rewarding solutions observed over all episodes of training. The Pareto Dominant solutions gathered during the 1000 episodes of training shows the models continuous improvement as it converges to the solution.

Pareto Frontier

$$\rho = 10/10 = 1$$

The small graph's final Pareto Dominant solution, Figure 4, is 10 which is equivalent to the solution derived from the analytical methods.

$$\rho = 52/49 = 1.06$$

The medium graph's final Pareto Dominant solution that is concluded at the end of the 1000 episodes for the medium graph is 52, which is 3 actions away from the analytical solution of 49 actions required to complete the CPP of the medium graph.

$$\rho = 196/174 = 1.126$$

The large graph's Pareto Dominant solution set concludes at the end of 1000 episodes with a policy that completes the CPP on the graph in 196 steps. This compares to the analytically derived solution of 174 steps with an approximation ratio of

$$\rho = 1.126$$

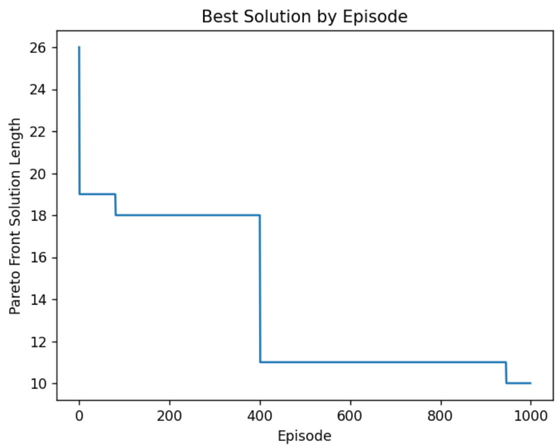

Figure 4: Pareto Front Best Policy by Episode Small Graph, optimal = 10

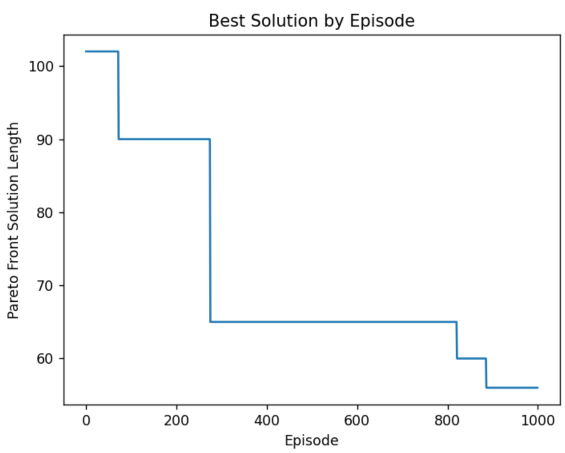

Figure 5: Pareto Front Best Policy by Episode Medium Graph, optimal = 49

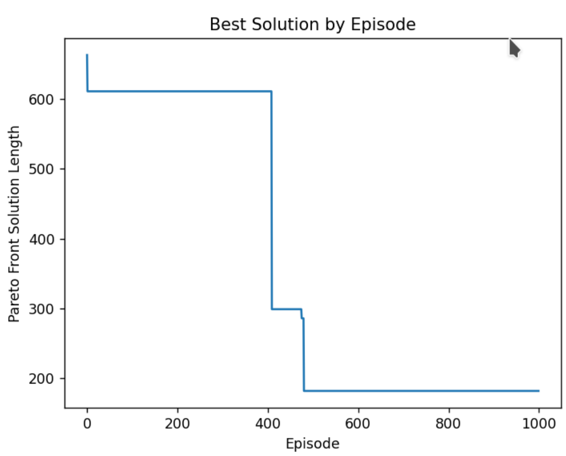

Figure 6: Pareto Front Best Policy by Episode Large Graph, optimal = 174

Cummulative Reward

The cummulative reward evaluation of a Multi-Objective RL model is slightly altered from the traditional RL cummulative reward analysis. This is exaccerbated by the Single policy method used which creates a greedy combination of the policies formed for each objective individually.

This is exhibited in the models cummulative reward as a very slow growth with a large amount of variance in the objective reward early on as the agent uses the initial epsilon greedy method to explore. The agent explores (ignores one objective in favor of the other) causing volatility in either objectives reward, and as the training continues the reward settles but stays moderately variable as the agent shift to a more predominantly exploitative action selection policy.

This requires the agent to synthesize a Pareto Dominant policy out from the maximum position in between the rewards of the two objectives. This combination of the policies in a greedy way leads to final policies that allow for the consideration of two objectives at the cost of focusing upon achieving a single objective.

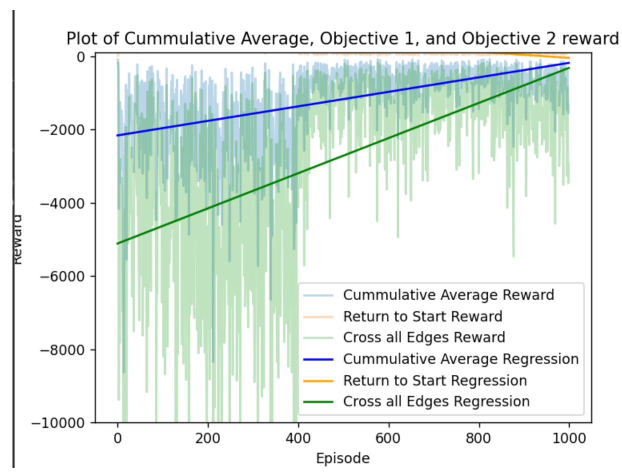

Figure 7: Reward by Objective and Average Small Graph

The cummulative reward of the agent over training upon the small graph trends to a higher reward. This indicates the agent is learning from the environment. The final policy being a greedy combination of the two objectives and learning from the Pareto frontier does not ensure that the final policy is optimal only that it was informed by the Pareto frontier. This exemplifies the way that the final policy must be formulated from two contrary policies. This still allows the agent to formulate policies that approximate analytically derived approximate solutions. The bootstrapping nature of this algorithm is slowed by the balancing of the two objectives. The final policy obtained from the algorithm upon this map is coherent and followable without taking into consideration the Pareto front. This is due to the algorithm having had learned from the Dominant solution and informed its Q values with such solutions.

The cummulative reward of the agent over training upon the medium graph trends to a higher cummulative reward,

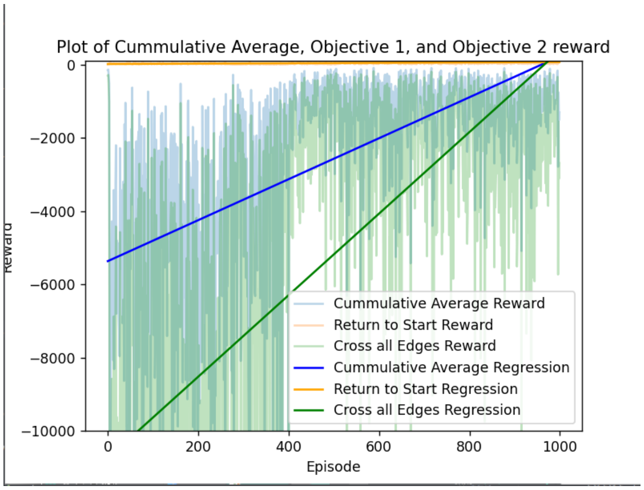

Figure 8: Reward by Objective and Average Medium Graph

the agents reward grows as the training progresses. This indicates that the single policy is being improved by training alongside the Pareto Frontier solutions as well as that the Pareto Frontier is expanded with exploration of the solution space. The stability of the solution is additionally stabilized as the variance of the two objective rewards is calming by the end of the training. The values stay highly variant throughout training indicating a high variance in the reward of each objective. This is likely due to the dichotomy of action selection caused by having multiple objectives. The is counteracted by utilization of the pareto front in action selection and policy formation. The final policy obtained from this algorithm upon this map is incoherent as it wishes to move from node 7 to 6 and then back to 7. This is remediated via incorporating the Pareto Dominant solution into policy formation. This additional logical requirement of upon the second visitation to node 7 to traverse to node 10 instead then creates a coherent policy that solves the CPP in approximately optimal steps.

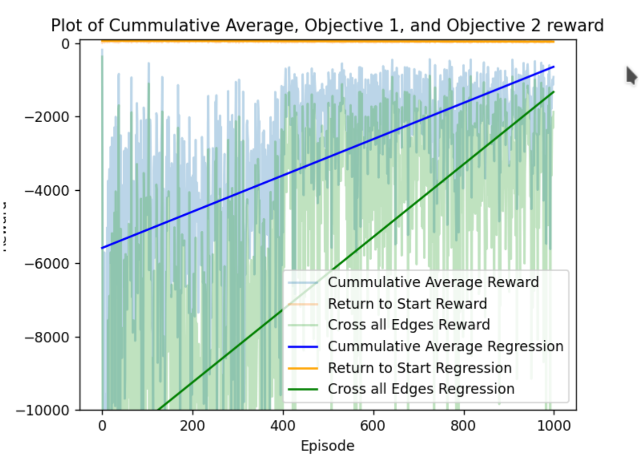

Figure 9: Reward by Objective and Average Large Graph

The cummulative reward of the agent when trained upon the large graph shows the continuous upward trend of the reward for the agent. This is balanced by high variance of the reward itself. Even though it is overtime trending to a higher reward, it does this at great computation cost. The training of this algorithm upon larger graph grows runtime significantly Although the algorithm has approximated an optimal solution for the CPP on the larger graph its solution is suboptimal and the computational costs are astronomically higher than the analytical solution computational costs upon the same graph.

The final policy obtained from this algorithm upon this graph is incoherent as it wishes to move to 3 trap oscillatory action circuits. These are overcome by the Pareto front method as well as it's insights allow for altering the policy so that when those nodes are come to upon the second visitation to utilize a different action selection method derived from the Pareto Dominant solution. This leads to the final policy that approximates the analytical solution. This approximate solution comes at great computation costs, which must be considered when applying this algorithm as the costs when compared to the analytical solution are substantial.

Summary of Results

The algorithm of Single Policy MORL learns to balance between the two objective rewards to find a golden mean policy between the two objectives separate optimal policies. As the agent learns, the volatility of average reward becomes lower as the Q values and exploitation rise ensuring the agent balances between the two objectives. This forms the single policy MORL.

The single policy MOQ is the best solution that incorporates both objectives. The optimal solutions for the single policy is the Pareto Dominant solution of the Pareto Frontier. The Pareto Frontier is saved as the best solutions that have been observed in all episodes of training. The Pareto Dominant solution is the solution of an observed episode of training that achieved both objectives in the lowest number of actions.

The fact that MORL requires far higher computational costs than the traditional algorithm must be considered when applying this method to real world problem graphs. The decreased optimality of the solutions when run for a short time must also be considered. The optimal solution may be found with this algorithm if it is ran for a very large number of episodes, but the traditional algorithms runtime being quite low creates a

The Chinese Postman Problem as approached by Multi-Objective Q-Learning Reinforcement Learning yields solutions to Chinese Postman Problems that are within the +/- 10% approximation goal of evaluation initially set forth. This accomplished the initial project goals, but did not exceed the goals as the runtime to solve the CPP using RL is an order of magnitude slower than using the traditional non-polynomial solution. The results shows that the algorithm finds the optimal solution for small graphs, and that when the algorithm is applied to larger problem graphs it is likely converge to at least the approximately optimal solution guaranteed by analytical methods; when the algorithm is let run for enough episodes. The sufficient number of episodes growing as the environment graph grows. This result indicates improvement must be made in the solutions for the application of MORL to larger instances of CPP to be a worthwhile endeavor for expanding the solution space of CPP. Preliminary results indicate potential in further investigation as these early results resemble analytical methods and further work could improve even further upon the results.

Conclusions

The research conducted in this experiment has been focused upon using Single Policy Multi-Objective Q-learning for application to the Chinese Postman Problem. The objectives of the agent are to cross all edges, and then to return to the start node. The application of Multi-Objective Q learning to complete the CPP's two objectives is accomplished. The formulatiion of a policy an agent utilizes to approximately optimally solve the CPP is accomplished via training Q values on many episodes. Additionally, informing the Q values of Pareto front solutions via learning from them improves correct action selection. The experiments result in approximate solutions that match analytical methods for solving the CPP by the initially proposed approximation ratio of $\rho = 1.1$.

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

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

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. *Deep reinforcement learning with double q-learning*. In Proceedings of the AAAI conference on artificial intelligence, volume 30.

Hayes, C.F., Rădulescu, R., Bargiacchi, E. et al. 2021. *A practical guide to multi-objective reinforcement learning and planning.* Auton Agent Multi-Agent Systems

Yang, R., Sun, X., Narasimhan, K. 2019. *A generalized algorithm for multi-objective reinforcement learning and policy adaptation*. Advances in neural information processing systems

Van Moffaert, Nowé, A. 2014. *Multi-objective reinforcement learning using sets of pareto dominating policies*. The Journal of Machine Learning Research

Gábor, Z., Kalmár, Z., Szepesvári, C. 1998. *Multi-criteria reinforcement learning*. ICML

Natarajan, S., Tadepalli, P. 2005. *Dynamic preferences in multi-criteria reinforcement learning*. In Proceedings of the 22nd international conference on Machine learning

Majumder, S., Kar, S., Pal, T. 2019. *Uncertain multi-objective Chinese postman problem*. Soft Computing

Acknowledgments

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