Reinforcement Learning for Adaptive Routing of Autonomous Vehicles in Congested Networks

Jonathan Cox
Aeronautics & Astronautics
Stanford University

Brandon Jennings
Mechanical Engineering
Stanford University

Steven Krukowski
Aeronautics & Astronautics
Stanford University

Abstract

The authors present a reinforcement learning approach for routing autonomous vehicles in a dynamic network with congestion. The policy is an extension of Q-Routing, a technique originally developed for packet routing in communication networks. The proposed algorithm is shown to perform as well as individual shortest path planning in static conditions and far outperforms shortest path planning in scenarios with road congestion.

1 Motivation

Cooperative routing has received significant attention due to its wide range of application in the control of unmanned aerial vehicles and distribution of packets in communication networks. This paper describes the application of a novel technique inspired by reinforcement learning to the cooperative control of a multivehicle network that outperforms shortest path algorithms in dynamic environments.

Optimal path planning for a single agent has been studied extensively in [1] and [2]. These algorithms provide superior performance in static networks with multiple agents but have limitations in dynamic networks, where the reward function at each state is either time or state-dependent.

The authors investigate the taxi routing problem, where each taxi is represented as an agent originating at some network node with the goal to traverse the network and arrive at a destination node. The state model was developed under the following assumptions: 1) There is a non-trivial ratio of autonomous agents (taxis) relative to independent agents (other vehicles) such that the time to traverse each edge in the network is dependent upon the number of autonomous agents in the immediate vicinity of that node and 2) The contribution to the state model by the independent agents can be represented as a static function dependent only on network location. These can be interpreted as a congestion model, where 2) is a base level of congestion due to the independent agents and 1) is the contribution to congestion by the autonomous agents.

In the spirit of reinforcement learning, the authors assume no prior knowledge of the congestion model. In this way, the routing algorithm is robust to sharp changes in congestion (traffic accidents, etc.) and can be applied to a new city network with no alteration.

2 Q-routing

2.1 Basic Update

The inspiration the for approach used by the authors is named Q-routing due to its similarity to the reinforcement learning technique Q-learning [3]. The Q function is a matrix mapping states and actions to rewards $S \times A \mapsto R$. The Q matrix holds at each entry $Q_x(d, y)$ the time estimated for an agent to traverse the network from node x to node d by way of neighbor node y. Upon traversing from node x to y, the agent updates $Q_x(d, y)$ as follows:

$$Q_x(d, y) := Q_x(d, y) + \alpha(s + t - Q_x(d, y)) \tag{1}$$

d - destination node
x - current node
y - neighbor node of x
α - learning rate
s - travel time between nodes x and y
t - $min_{z \in \{neighbors\ of\ y\}} Q_y(d, z)$

When a taxi arrives at a node, it immediately uses the information about the length of time on that leg of its trip to update the previous nodes estimate $Q_x(d, y)$. A table of estimates is kept for each node and turn direction to each possible destination node. After the update is applied, the next action is determined by finding the optimal decision at that node, $argmin_{y \in neighbors\ of\ x} Q_x(d, y)$.

2.2 Extended Update

The authors propose an extension to Q-routing, where each taxi stores its path history in memory and uses this information to update previous nodes. This memory is cleared when the taxi reaches its destination and the process repeated. The goal of this extended update is to accelerate convergence to an optimal policy in static conditions.

When a taxi arrives at a node, each node in its history is updated based on the current estimate of the travel time to destination node from that node in the taxi history, the time it took the taxi to get from that node to the current node, and the estimated time to go from the current node to the destination.

$$Q_{x^i}(d, y^i) := Q_{x^i}(d, y^i) + \alpha(p_i + min_{y \in \{neighbors\ of\ x^j\}} Q_{x^j}(d, y) - Q_{xi}(d, y^i)) \tag{2}$$

i - element in path history, $i \in \{1, ..., j\}$
j - last element in path history
p_i - travel time between nodes i and j

A comparison of Q-routing, the proposed Extended Q-routing, and the shortest path algorithm A* for moderate congestion is shown in Figure 1.

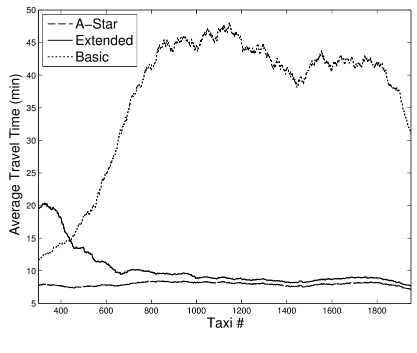


Figure 1: Comparison of Q-routing Basic and Extended

3 Simulation

A simulation was developed to test the performance of the algorithm. The test map is a small representation of the city of Manhattan, appropriately named 'Mini-Manhattan' (see Figure 2). Mini-Manhattan is a 10×3 node network with two missing nodes to account for Central Park. Each edge, or road connecting nodes, was chosen to be one mile for simplicity. The maximum velocity allowable on each road, or speed limit, was chosen to be either 45 mph (light green), 30 mph (dark green) or 20 mph (yellow) based on characteristics of real New York City streets. Customers are assumed to be picked up at street corners (nodes). The locations of taxis are propogated based on a finite time step, where the velocity is assumed to be constant for that step. When a taxi arrives at a node, it decides which direction to turn based on its current routing algorithm.

The Q matrix is initialized at every entry to the Manhattan distance between nodes. In this way, the taxis have only information regarding road geometry and must learn the speed limits through trial and error.

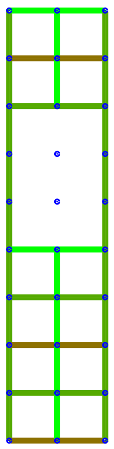


Figure 2: Mini-Manhattan Test Map

3.1 Congestion Function

Many models have been developed to relate traffic and vehicle throughput, yet no single model has been proven to accurately depict this relationship for all road types and geometries [4]. The authors utilize a modified sigmoid function to represent the relationship between number of vehicles on a road and mean velocity

$$V(x) = V_{max}\frac{(1+e^{\beta-\mu})(e^{a-\beta x})}{1+e^{\mu-\beta x}} \tag{3}$$

where x is the number of cars on a road and β and μ are constants.

This representation is advantageous in simulation because it satisfies the conditions $V(1) = V_{max}$ and $\lim_{x\to\infty} V(x) = 0$. The minimum velocity was bounded to ensure that every vehicle completed its route in a finite time.

The choice of congestion function is not critical to demonstration of the performance of the algorithm. Rather, the difficulty in deriving a meaningful relationship between distribution of vehicles and mean velocity for every road type and geometry is a motivating factor for use of a learning algorithm.

3.2 A* Search

The A* search algorithm was used as a baseline model against which to evaluate the performance of Q-routing. A* is used for finding the shortest path between two points in a node network with edge costs. It is an extension of Edsger Dijkstras 1959 algorithm [2] that achieves faster performance through the use of heuristics.

In the baseline scenario, a taxi calculates its A* path from origin to destination upon receiving a customer. This process is done independently, based on knowledge of the speed limits and road geometry. The taxi then stores this path in memory and follows it to the destination. In the baseline scenario, the taxis are given knowledge of all static parameters of the system, but not the locations of other taxis or the congestion function.

4 Results

4.1 Test Case 1: No Congestion

We considered three test cases to show the superiority of the proposed algorithm to shortest path planning with A*. In the first case, both algorithms are run on the test map with no effect on velocities due to congestion. Figure 3 shows that after an initial learning period, Q-routing average travel times converge to those of A*. The taxi network running the Q-routing algorithm is able to learn the speed limits through trial and error.

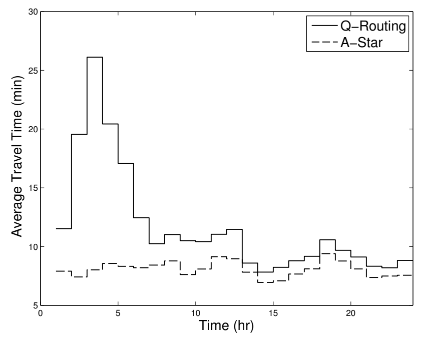


Figure 3: Test Case 1: Average Travel Times

4.2 Test Case 2: Unidirectional Travel

In the second test case, taxis pick up customers at the six southernmost nodes on the map and deliver them to the most northeast node. This is a reasonable model for morning or evening traffic, as commuters travel to and from work. Figure 4(a) shows that the taxi network running Q-routing is able to outperform the taxi network running A* in average travel time.

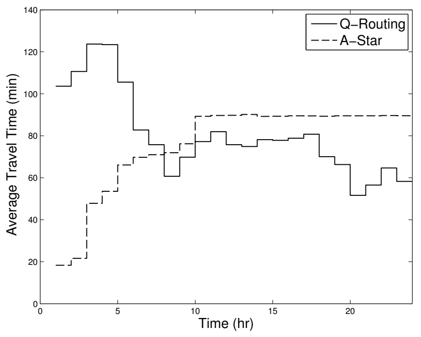
A primary concern in using this algorithm for vehicle routing is sacrificing individual performance in order to minimize group travel time. It is unrealistic to expect a single taxi to vastly increase its own travel time in order to save a small fraction of time for many taxis. Figure 4(b) shows that no taxi increases its own travel time by more than 50 percent after the initial learning period in this scenario.

4.3 Test Case 3: Game Day

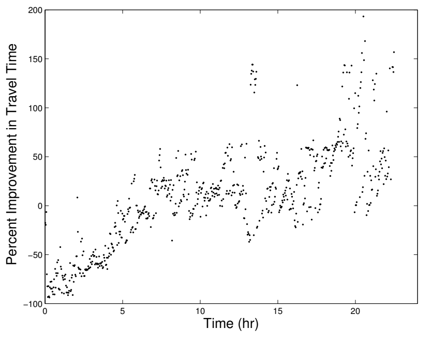
In the third test case, the base customer arrival and destinations were randomized and arrival rate set constant. In hour 20, the arrival rate was doubled, with half the arriving taxis traveling to one node. This scenario simulates a sporting or other large event that causes an influx of traffic. Figure 5 shows the average travel times for this case. Q-routing average times increase at the start of the traffic influx but quickly adjust to outperform those of A*.

5 Conclusion

An extension of Q-routing was shown to outperform A* in multivehicle routing on a traffic network with congestion. The algorithm is ideally suited to solution of this problem due to the dynamic nature of traffic and difficulty of modeling congestion on roads.



(a) Average Travel Times



(b) Improvement in Travel Time with Q-routing

Figure 4: Test Case 2

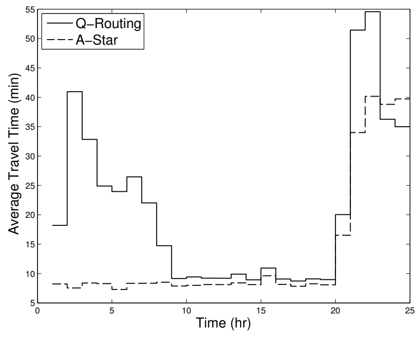


Figure 5: Test Case 3: Average Travel Times

Several simplifying assumptions were made in this analysis. Adding variation to road geometry and allowing customers to arrive at edges would result in a more realistic simulation, although the authors submit that these changes would only prove to highlight the benefits of Q-routing.

The trade-off between individual performance and overall system performance is a topic that warrants further exploration. In the results presented in this paper, there was not a significant degradation in individual travel times to achieve shorter average travel times. However, a metric that quantifies this trade-off is important to avoid the sociological implications of passengers never reaching their destinations.

A downside to this solution approach is that Q-routing suffers from the curse of dimensionality. The Q matrix grows at $O(n^2)$ with number of nodes. An approach to remedying this is to partition a large city into subspaces, such that the Q matrix contains travel time information for every node in its subspace and from its own subspace to others. The authors intend to explore this approach in future work.

References

[1] On a routing problem, R. Bellman, 1958, Quarterly of Applied Mathematics 16:87-90.

[2] A note on two problems in connexion with graphs, E.W. Dijkstra, 1959, Numerische Mathematik 1:269-271.

[3] Packet Routing in dynamically changing networks: a reinforcement learning approach, J. Boyan and M.L. Littman, 1994, Advances in Neural Information Processing Systems 7:671-678.

[4] Traffic stream characteristics, N.H. Gartner, C.J. Messer and A.K. Rathi, 1996, Traffic Flow Theory, United States Federal Highway Administration.