Reinforcement Learning Routing in Transportation Network: A review and comparison with a case study

Hoang Dat Pham
Electrical and Computer Engineering
University of Manitoba
Email: phamhoad@myumanitoba.ca

Sharath Narasim
Civil Engineering
University of Manitoba
Email: sharath.narasimhamurthy@umanitoba.ca

Babak Mehran
Civil Engineering
University of Manitoba
Email: babak.mehran@umanitoba.ca

Ahmed Ashraf
Electrical and Computer Engineering
University of Manitoba
Email: ahmed.ashraf@umanitoba.ca

***Abstract*—Finding the shortest path in a network is a classical problem, and many search strategies have been proposed to solve it. This paper investigates the applicability of some benchmark search strategies in a simulated transportation network where link costs are dynamically estimated with vehicle mean speeds, and compares their performance with a reinforcement learning routing algorithm, which was originally used in a computer network, when it is re-purposed in the same transportation network.**

***Index Terms*—Shortest Path, Reinforcement Learning Routing, Traffic Network, Dijkstra, A*, Dynamic Link Cost.**

I. Introduction

Autonomous transportation will soon be used as a means of public transportation in the near future. Such a transportation system demands that estimating the shortest path be well-adapted to the continuously changing nature of the traffic network. In this stochastic network, the link cost (e.g. travel time) of a transportation network is dynamically changing and influenced by many traffic factors such as traffic congestion, road work, bad weather, etc. In this context, autonomous transit is integrated with a real-time routing system that can estimate the shortest path before the trip and reroute the vehicle in response to fluctuating link costs.

Over the past decades, the shortest path problem has been extensively investigated with applications ranging from computer networks to transportation networks and many approaches have been explored to examine the effectiveness of search strategies. In the transportation network, traditional search strategies have been thoroughly examined, which are uninformed search, informed search, and incremental search [1], [8], [11]–[14], [19]–[22]. However, there is another search strategy, namely, reinforcement learning search [2], which is primarily applied in computer networks but has not been investigated in transportation networks. Moreover, reinforcement learning studies have been focused in recent years on transportation, and numerous studies on reinforcement learning routing [5] have been proposed for vehicle routing optimization, but a practical comparison in terms of optimal shortest path between the reinforcement learning approach and traditional approaches should have been investigated.

The main contributions of this paper are summarized as below:

- We investigate the performance of some shortest path algorithms that represent traditional search strategies in simulated transportation network where link costs are dynamically estimated and highly correlated based on the current traffic condition.
- We investigate the use of the reinforcement search strategy in the same transportation network and compare its performance to that of other search strategies.

In this paper, a transportation network is simulated with VISSIM [6] and traffic obstructions are introduced into the network to induce fluctuations in vehicle volumes and speeds. These vehicle speeds are converted into link costs and are used as weights for shortest path algorithms.

II. Related Work

The problem of shortest path estimation, which has been widely researched, is a principal and classical challenge in transportation and computer networks. Numerous publications in related field have been proposed to improve the criteria that validate a good algorithm for finding an optimal path in a network. All shortest path algorithms which are investigated in this paper are within the context that the transportation network is a directed graph with link cost as travel time and is non-negative. Shortest path algorithms can be evaluated based on the following four criteria:

• Completeness: This criterion determines whether or not the algorithm is guaranteed to find the solution to the problem, if one exists.

• Optimality: This criterion evaluates whether the solution provided by the algorithm is the best.

• Time complexity: This criterion evaluates how long the algorithm takes to solve the problem. Usually, it is expressed in a big O time complexity as a degree of number of inputs.

• Space complexity: this criterion evaluates how much memory space the algorithm consumed to reach the final solution.

A. Search Strategies

Traditional AI literature [18] distinguishes three types of search strategies:

- Uninformed search, in which the algorithm employs no method of estimating how close the search process is to a destination.
- Informed search, in which the algorithm employs heuristics to direct the search to its destination.
- Incremental search, in which the algorithm reuses information from previous searches to find updated shortest path solutions faster than searching for the shortest path from the beginning.

However, there is another type of search strategy based on reinforcement learning that estimates the shortest path by exploring the transportation network, and estimated travel time is given as a form of reward.

1) Uninformed Search: Uninformed search strategies refer to a group of search techniques known as searching without any knowledge to reach the objective. It is also known as blind search. Table I summarizes the current uninformed search techniques [18], which are breadth-first search, depth-first search, depth-limited search, iterative deepening search, uniform cost search, bidirectional search, Dijkstra [4], and their criteria performance. In Table I, the depth-first search is complete only with graph search, which is applicable to transportation network. Hence, Dijkstra is considered as a benchmark uninformed search algorithm because it utilizes no heuristic function to direct the searching process, but its performance surpasses the other algorithms in the same algorithm class.

Uninformed Search	Completeness	Optimality	Time Complexity	Space Complexity
Breadth-first	Yes	Yes	$O(b^d)$	$O(b^d)$
Depth-first	Yes	No	$O(b^m)$	$O(bm)$
Depth-limited	No	No	$O(b^l)$	$O(bl)$
Iterative Deepening	Yes	Yes	$O(b^d)$	$O(bd)$
Uniform cost	Yes	Yes	$O(b^{l+[C/e]})$	$O(b^{l+[C/e]})$
Bidirectional	Yes	Yes	$O(b^{d/2})$	$O(b^{d/2})$
Dijkstra	Yes	Yes	$O(n^2)$	$O(n^2)$

b is branching factor, *d* is depth of the solution, *m* is maximum depth of tree, *l* is depth limit of tree, *C* is cost of optimal solution, *e* is each step closer to goal node and *n* is number of nodes

Table I
UNINFORMED SEARCH [18].

2) Informed (Heuristic) Search: The primary benefit of heuristic strategies is that the search scale can be limited. Many shortest path algorithms with a limited search area have been proposed, such as best-first search [15], greedy best-first search, hill climbing search, and A* [7] as in Table II. These types of searches attempt to reduce search efforts by utilizing various sources of additional information. Also, the A* algorithm is complete and optimal only if its heuristic function is admissible and monotonic, and many modern algorithms such as D* lite [10] and LPA* [9] are based on A*. Therefore, the A* algorithm stands out in the class of informed search because of its ability to complete the search process and produce optimal results.

Informed Search	Completeness	Optimality	Time Complexity	Space Complexity
Best-first	No	No	$O(b^m)$	$O(b^m)$
Greedy best-first	No	No	$O(b^m)$	$O(b^m)$
A*	Yes[2]	Yes[2]	$O(b^d)$	$O(b^d)$
Hill Climbing	No	No	$O(\infty)$	$O(b)$

Table II
INFORMED SEARCH [18].

3) Incremental Search: In a transportation network where there are only a small number of link costs that change, therefore, the changes only affect a small fraction of the graph. As a result, computing the entirely new shortest path is not necessary but to update the fraction of the graph that has been affected by link cost changes. The incremental search methods are used to solve dynamic shortest path problems, which require determining shortest paths repeatedly as the topology of a graph or its link costs only change partially. One of the incremental search algorithms is the Ramalingam and Reps' algorithm [16] (RR), also has another name as the DynamicSWSF-FP algorithm. However, if all the link costs in the graph change, the incremental search algorithms can not benefit from previous search information.

DynamicSWSF-FP Algorithm: Only a portion of links in dynamic transportation networks change their cost between updates. Some nodes' start costs remain unchanged and do not need to be recalculated. This implies that re-computation of the optimal route may be wasteful because some of the previous search results may be reused. Incremental search methods, such as the RR algorithm, reuse information from previous searches to find the shortest paths for a series of similar path-planning problems faster than solving each path-planning problem from scratch. The issue with reusing previous search results is determining which start costs have been affected by the cost update operation and must be recalculated. The RR algorithm employs two estimates: one that corresponds directly to start distance in Dijkstra's algorithm and another one is a right-hand-side value (*rhs*) for checking the local consistency to prevent path recalculation.

4) Reinforcement Learning Search: The field of reinforcement learning has grown significantly over the past decades, with applications ranging from robotics to game AI. One of the algorithms based on reinforcement learning that was proposed primarily for computer networks that can be used in transportation systems is Q-routing [2]. This reinforcement learning algorithm is implemented in computer networks where information packets are sent to experience transmission

Incremental Search	Completeness	Optimality	Time Complexity	Space Complexity
DynamicSWSF-FP [17]	Yes	Yes	$O(\|\|\delta\|\| \cdot (log\|\|\delta\|\| + M\delta))$	Not available

|| δ || is a measure related to "the size of the change in the input and output", $M\delta$ is a bound on the time required to compute the function associated with any vertex in *Changed U Succ(Changed)* [17]

Table III
INCREMENTAL SEARCH.

time and there was no investigation of the algorithm in transportation network.

Q-routing Algorithm: Q-routing was proposed by Boyan and Littman based on Q-learning. The Q value is the estimated link cost of the whole path. At any node, the Q-routing algorithm explores the shortest path by deciding the next node that minimizes the link cost from the next node to the destination node, and updates the estimated Q value for the current node:

$$Q_x(d,y) = Q_x(d,y) + \eta(\underset{y \in N(y)}{min} Q_y(d,z) + t - Q_x(d,y)) \quad (1)$$

where d is the destination, y is the next node, z belongs to the list of the adjacent nodes of y, η is the step-size, and the reward t is the link cost from x to y.

A Q-table, which comprises Q values, is initialized with zeros. During the first few training loops, the Q-routing acts as an uninformed search algorithm as it attempts to fill all zero values with estimated link cost. In subsequent loops, Q-routing performs as an informed search algorithm because it chooses the next node that minimizes the previous estimated link costs to destination.

Reinforcement Learning Search	Completeness	Optimality	Time Complexity	Space Complexity
Q-routing	Yes	Yes	$O(T * n^2)$	$O(n^2)$

T is the number of training loops

Table IV
REINFORCEMENT LEARNING SEARCH

III. TRAFFIC NETWORK SIMULATION

A. Traffic Network Description

A simple transportation network in the form of square grids is used in this study as shown in Figure 1(a). The network has 25 nodes and 80 links (each 2-lane and directed). The length of each link is 2500 meters. The network has 9 signalized intersections. The signal heads are shown in Figure 1(b) and (c). Right turning movements are considered free. Through and left turning movements are simultaneously allowed (but controlled by the traffic signal). The node annotations and coordinates are shown in Figure 2. The four signal heads at an intersection are sequentially operated with a cycle length of 240 seconds (57s green + 3s amber per signal head). All the 9 traffic signals are synchronized.

The demand for travel arises at two main origin intersections, O1-O2 and O3-O4, which are located at node 1 and 5. Two main destination intersections, D1-D2 and D3-D4, which are located at node 25 and 21 are considered. Coordinates of

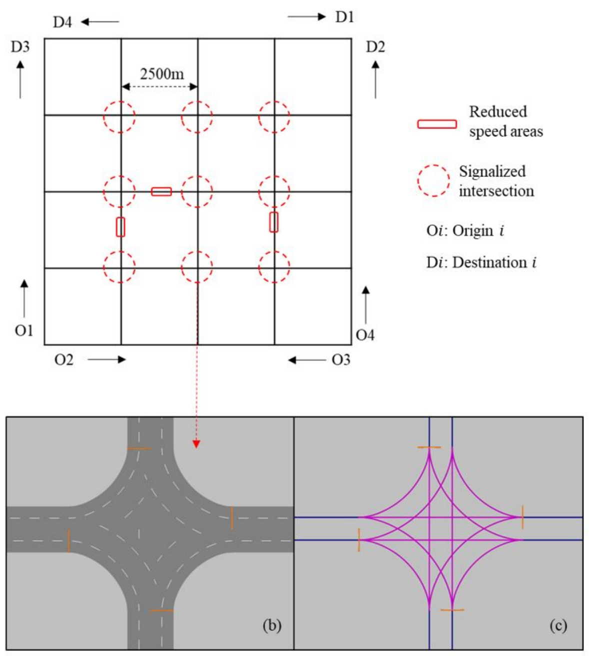

Figure 1. Details of the toy network used in the study

each intersections of origin and destination can be viewed in Figure 2. The magnitude of travel demand between OD pairs i,j ($Demand_{i,j}$ ∀i,j=1,2 "and" i≠j) is taken as 500 vehicles per hour. Such a large value of travel demand is considered to ensure a build-up of queue/congestion in the network. Beside main OD pairs that have large travel demand, other random OD pairs are also used to run algorithms.

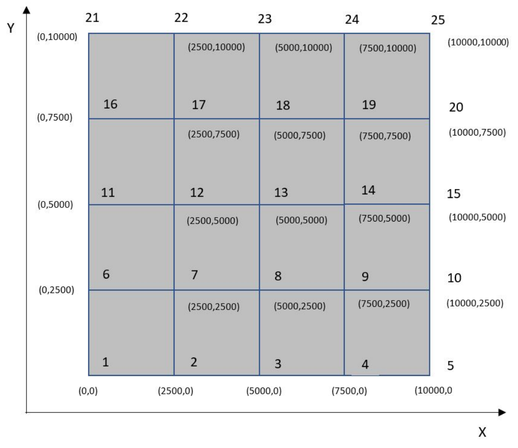

Figure 2. Node numbers and coordinates

The aforementioned network is modeled in PTV VISSIM, which is a microscopic traffic simulator. As the size of a network increases, it becomes very tedious to determine and assign routes (sequence of links) for vehicles between several origin-destination pairs. Also, a predetermined route assignment in a simulation study does not reflect the route choices made by drivers in the real world [6]. Hence, the route choices are dynamically made in this study. The ability of VISSIM to compute dynamic stochastic user equilibrium is exploited to determine the routes dynamically. A comprehensive representation of route choice is thus possible. The vehicle composition of 90% cars and 10% heavy vehicles is used. 98% of the vehicles are passenger cars and the rest are heavy goods vehicles. The default speed distributions and driving behaviors utilized in this network simulation are : the desired speed distribution was as shown in Figure 3, the Wiedemann-74 car-following driving behavior is used as it is considered suitable in urban environment, and the default Car-following parameters used are:

- Average standstill distance: 2m
- Additive part of safety distance: 2
- Multiplicative part of safety distance: 3

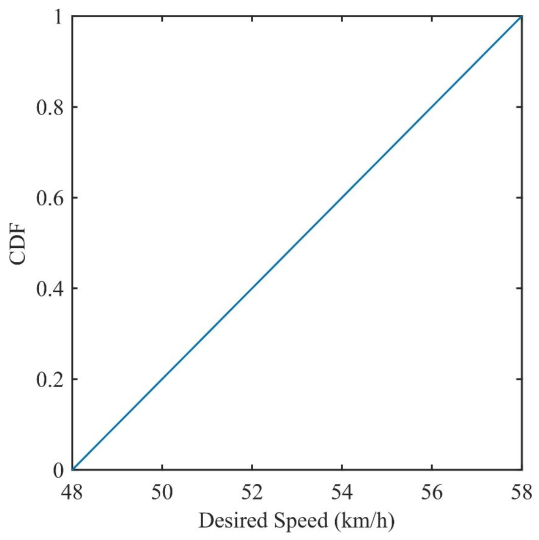

Figure 3. Speed Distribution

VISSIM also has a feature termed "reduced speed areas", which is used to create temporary or localized congestion (e.g., a traffic incident). Three links are randomly chosen to introduce congestion as shown in Table 5. A simulation step size of 0.1s and a simulation period of 2 hours are adopted. After every 100 simulation steps, the state of every vehicle in the network is sampled (and stored for further analysis) by using VISSIM COM. The state of a vehicle includes the link on which a vehicle is plying, position on that link, lane occupancy, instantaneous speed, acceleration, and vehicle type. Furthermore, the simulation input parameters are as provided above. Normally, "default behaviors" are mentioned [3].

Link	Length of congestion zone (m)	Desired speed (km/h)
26	500	20
10	200	12
34	300	12

Table V
CONGESTED LINKS

The desired speed distribution within the congestion zone is as shown in Figure 4.

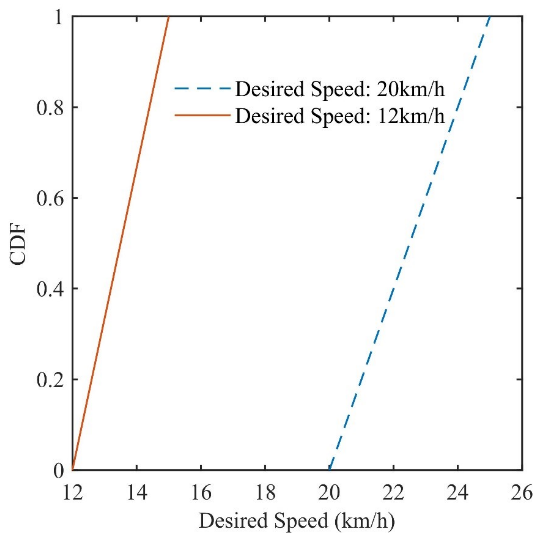

Figure 4. Speed distribution within the congestion zone

B. Dynamic travel time estimation

The objective is to understand the spatial and temporal variation of link costs (travel time). The state of all the vehicles in the network is extracted from Vissim after every 100 simulation steps. Every link between nodes i and j is considered to be composed of n segments (need not be of equal length) as shown in Figure 5. The length of the k^{th}segment of link ij is $l_{k,i,j}$ and total length of the link is L_{ij}.

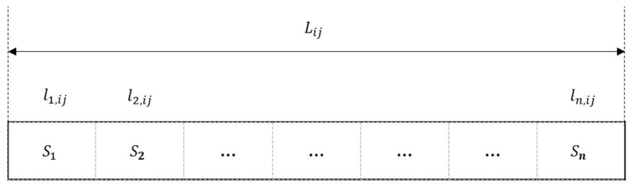

Figure 5. Segmentation of a link ij

The space mean speed of the segment k at a time step $t(SMS^t_{k,ij})$ can be computed from the extracted states. The cost of traveling on link ij at a time step t is then computed as (2):

$$C^t_{ij} = \sum_{\forall k} w_{k,ij} \times SMS^t_{k,ij} \qquad \forall ij,t \quad (2)$$

where, $w_{k,ij} = \frac{l_{k,ij}}{L_{ij}}$ is the weight for segment k.

IV. ALGORITHM PERFORMANCE

A. Algorithm Selection

From related literature, it is clear that Dijktra and A* are the benchmark candidates that represent uninformed search and informed search respectively in terms of performance criteria.

On the other hand, link costs in our simulated networks are estimated based on the number of link vehicles' speeds. As the number of vehicles that enter the link changes and vehicle speeds vary, link costs dynamically change at every query time for all links.. Therefore, the RR algorithms and similar incremental algorithms such as Lifelong planning A* [9] and D* lite [10] based on it cannot benefit from incremental search because the estimate *rhs* changes dynamically and, thus, local consistency will never be met. Therefore, in the context of our simulated network, there is no incremental search benefit and the RR algorithm is merely another version of Dijkstra's algorithm.

Hence, we chose some of the basic algorithms that best represent their classes, namely, Dijkstra for uninformed search, A* for informed search, and Q-routing for reinforcement learning, and compare their performances in our case study network. Figure 6, 7 and 8 show the algorithm. The heuristic function that we use in the A* algorithm is a Euclidean distance function that estimates how close the current node is to the destination based on the nodes' coordinates.

```
function Dijkstra(Graph, source):
   create node set Q
   for each node v in Graph:
      dis[v] ← ∞
      prev[v] ← UNDEFINED
      add v to Q
   dist[source] ← 0
   while Q is not empty:
      U ← node in Q with min dist[u]
      remove u from Q
      for each neighbor v of u:
         alt ← dist[u] + travel_time(u,v)
         if alt < dist[v]:
            dist[v] ← alt
            prev[v] ← u
   return dist[ ], prev[ ]
```

Figure 6. Dijkstra's algorithm

```
function Qrouting(Graph,source,destination):
   initialize Qtable = 0
   for each training step:
      current_node x←s
      take_node y←abitrary
   while current_node x is not destination d:
      take_node y=argmin(Q(x,d))
      take_next_node y'=argmin(Q(y,d))
      Q(x,y)=Q(x,y)+lr*(Q(y,y')+travel_time(x,y)-Q(x,y))
   return Qtable
```

Figure 7. Q-routing algorithm

B. Algorithm Performance

Static Network: We first test the Dijkstra and A* algorithms on our simulated transportation when there is no vehicle. The purpose for this test is to check the performance of the heuristic function in A*. When there is no traffic in the network, the link costs are the length of the links and the network is static. Figure 9 shows the equal performance between the A* and the Dijkstra algorithm for 10 OD pairs of (1,25), (5,21), (10,16), (6,24), (22,4), (4,16), (18,5), (12,20), (17,5) and (22,9).

Dynamic Network: when traffic is introduced into the simulated network, the network becomes dynamic and link

```
function A_star(Graph, source,destination):
   openlist = []
   closedlist = []
   add start node s to openlist
   f(s)=0
   while openlist is not empty
      currentnode = node with min f
      remove currentnode from openlist
      add currentnode to closedlist
      if currentnode is destination
         return reconstruct_path
      let children of currentnode = adjacent nodes
      for each child in children
         if child is in the closedlist
            continue to beginning of for loop
         child.g=currentnode.g + travel_time(child,currentnode)
         child.h=travel_time(child,destination)
         child.f=child.g+child.h
      if child.posiition is in openlist's node positions
         if child.g > openlist node's g
            continue to beginning of for loop
      add child to openlist
```

Figure 8. A* algorithm

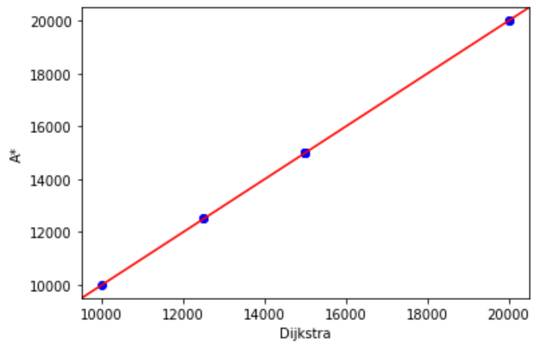

Figure 9. Quantile plot distance-based A* and Dijkstra with 10 OD pairs

costs are estimated using methods described in section 3.2. Dijkstra, A* and Q-routing are tested in this dynamic network.

Q-routing parameters are found to be optimal with 0.29 for learning rate and 200 for number of iterations. Figure 10 shows a graph with route cost for OD =1,20 versus learning rate, and route cost starts to be minimal when learning rate is at 29×10^{-2}.

In the dynamic simulated network, A*, Dijkstra, and Q-routing are performed every 10 seconds of simulation with all OD pairs, and a few OD pairs are selected for illustrating and comparing route costs. Figure 11 -14 shows the cost in terms of travel time of the entire route for OD = (1,25), (5,21), (4,16) and (6,14), and the data points for Q-routing overlap with Dijkstra. Moreover, Figure 25 illustrates the quantile plot of route costs between A*, Q-routing and Dijkstra for 10 OD pairs: (1,25), (5,21), (10,16), (6,24), (22,4), (4,16), (18,5), (12,20), (17,5), (22,9). Such OD pairs are selected to compare route costs because of the large, fluctuating travel demand in the vicinity of the links.

From Figure 11 to 25, the A* algorithm produces non-optimal shortest routes, resulting in large route costs, which are always larger than with Dijkstra and Q-routing. Furthermore, the heuristic function for A*, which is a Euclidean distance-based function, used in this simulated network is not

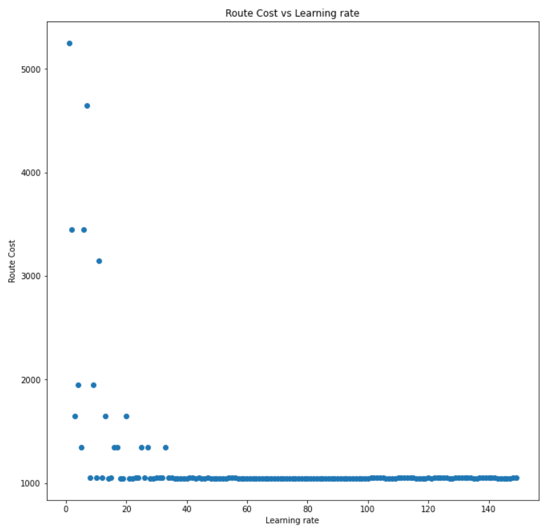

Figure 10. Route cost and learning rate for Q-routing

admissible when link costs are dynamic travel time. At any node during the A* run-time, the heuristic function always overestimates the route cost to the destination.

On the other hand, Q-routing delivers comparable route costs to Dijkstra during the entire simulation of the network. Figure 26 shows the overlapped routes between Q-routing and Dijkstra and the different routes of A*. In addition, the Q-routing algorithm is able to adjust the route to reflect the dynamic link cost to the destination, as shown in Figure 27.

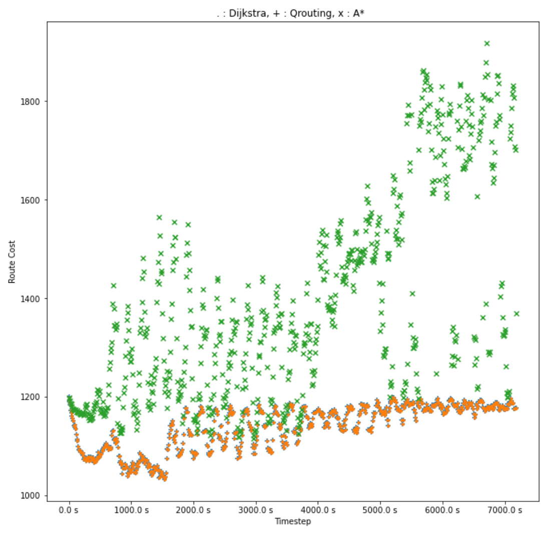

Figure 11. Route cost for OD = 1,25 with Q-routing (orange '+') , Dijkstra (blue 'o'), and A*(green 'x').

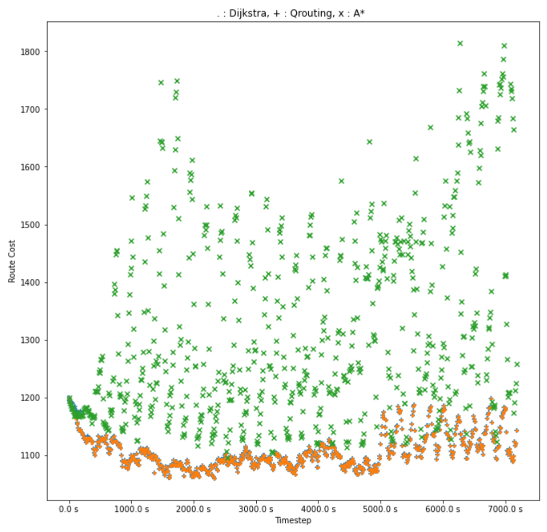

Figure 12. Route cost for OD = 5,21 with Q-routing (orange '+') , Dijkstra (blue 'o'), and A*(green 'x').

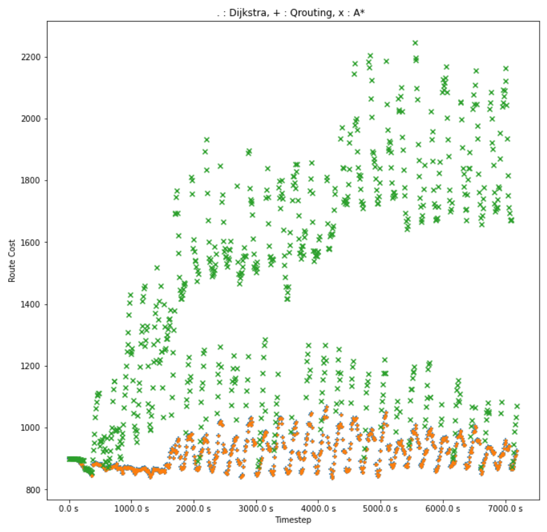

Figure 13. Route cost for OD = 4,16 with Q-routing (orange '+') , Dijkstra (blue 'o'), and A*(green 'x').

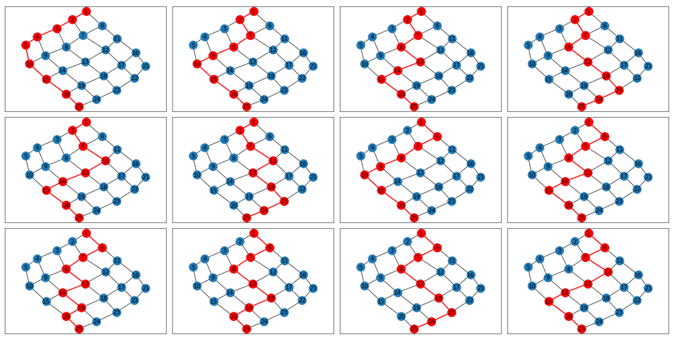

Figure 27. New shortest paths (red lines) with Q-routing when travel time estimation changes for a single OD = 1,25

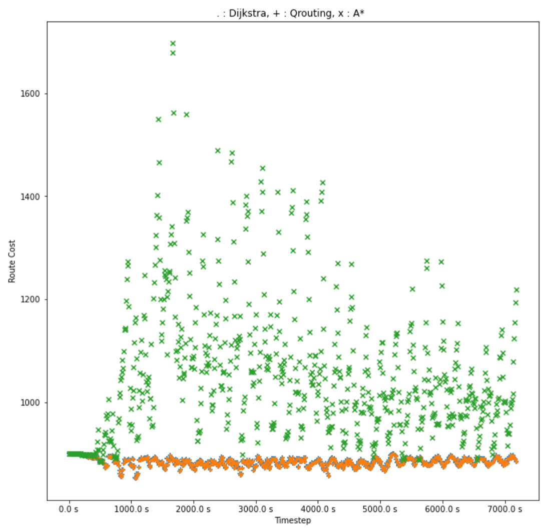

Figure 14. Route cost for OD = 6,24 with Q-routing (orange '+') , Dijkstra (blue 'o'), and A*(green 'x').

V. Conclusions

In this paper, we investigated the reinforcement learning search strategy for finding the shortest path in a simulated transportation network and compared it with the performances of two benchmark algorithms from traditional search strategies with dynamic link costs. Therefore, Q-routing can be considered as an alternative optimal algorithm for finding the shortest path in a simulated transportation network. For future work, a graph neural network can be integrated into the reinforcement learning routing to estimate shortest paths directly from network features such as traffic counts and link characteristics, rather than using link costs estimated from mean speeds as demonstrated in this paper.

References

[1] Q. Wu B. Huang and F. B. Zhan. A shortest path algorithm with novel heuristics for dynamic transportation networks. *International Journal of Geographical Information Science*, 21(6):625–644, 2007.
[2] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: a reinforcement learning approach. *Advances in Neural Information Processing Systems*, pages 671–678, 1994.
[3] H. Sebastian Buck, Nicolai Mallig, and Peter Vortisch. Calibrating vissim to analyze delay at signalized intersections. *Transportation Research Record*, 2615(1):73–81, 2017.
[4] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
[5] Nahid Parvez Farazi, Bo Zou, Tanvir Ahamed, and LimonBarua. Deep reinforcement learning in transportation research: A review. *Transportation Research Interdisciplinary Perspectives*, 2021.
[6] M. Fellendorf and P. Vortisch. Microscopic traffic flow simulator vissim. *Simulation International Series in Operations Research & Management Science*, pages 63–93, 2020.
[7] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics*, pages 100–107. IEEE, 1968.
[8] Pooja R. Katre and Anuradha Thakare. A survey on shortest path algorithm for road network in emergency services. *International Conference for Convergence in Technology (I2CT)*, 2017.
[9] Sven Koenig and Likhachev. Incremental a*. *Maxim*, 2001.
[10] Sven Koenig and Likhachev. D* lite. *Maxim*, 2002.
[11] D. Sunb L. Fua and L.R. Rilett. Heuristic shortest path algorithms for transportation applications: State of the art. *Computers and Operations Research 33*, pages 3324–3343, 2006.
[12] A. Madkour, W.G. Aref, F. U. Rehman, M. A. Rahman, and S. Basalamah. A survey of shortest-path algorithms. May 2017.
[13] Kairanbay Magzhan and Hajar Mat Jani. A review and evaluations of shortest path algorithms. *International Journal of Scientific and Technology Research*, 2(6), 2013.
[14] Stefano Pallottino and Maria Grazia Scutella. Shortest path algorithms in transportation models: Classical and innovative aspects. *Equilibrium and Advanced Transportation Modelling*, pages 245–281, 1998.
[15] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.
[16] G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21(2):267–305, 1996.
[17] G. Ramalingam and T. Reps. On the computational complexity of dynamic graph problems. *Theoretical Computer Science 158*, 16(1):233–277, 1996.
[18] Stuart J. Russell and Norvig Peter. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.
[19] Chanchal Dudeja Sunita Kumawat and Pawan Kumar. An extensive review of shortest path problem solving algorithms. In *Proceedings of the Fifth International Conference on Intelligent Computing and Control Systems*, 2021.
[20] Thorat Surekha and Rahane Santosh. Review of shortest path algorithm. *International Research Journal of Engineering and Technology*, August 2016.
[21] F. B. Zhan and C. E. Noon. Shortest path algorithms: An evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998.
[22] F. Benjamin Zhan. Three fastest shortest path algorithms on real road networks: Data structures and procedures. *Journal of Geographic Information and Decision Analysis*, 1(1):70–82, 1997.

Appendix

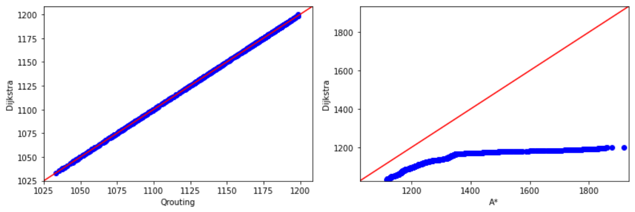

Figure 15. OD=(1,25)

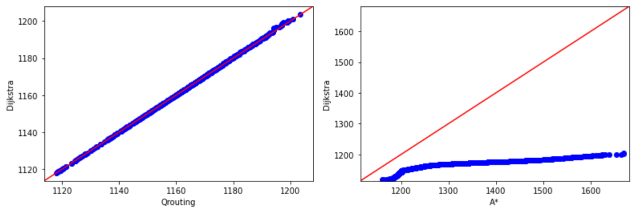

Figure 16. OD=(5,21)

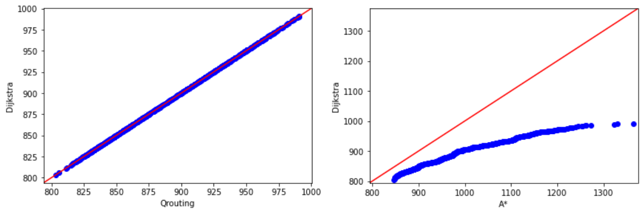

Figure 17. OD=(10,16)

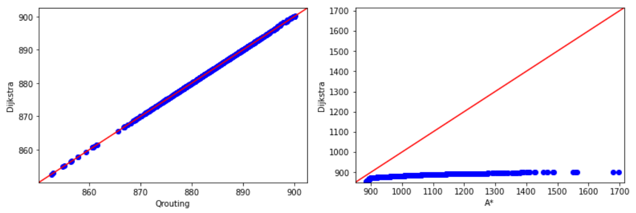

Figure 18. OD=(6,24)

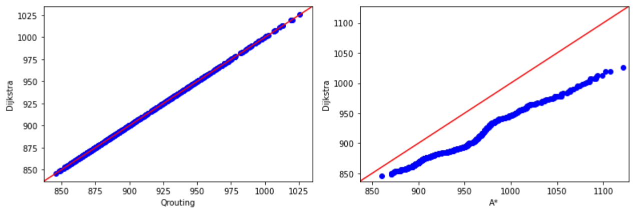

Figure 19. OD=(22,4)

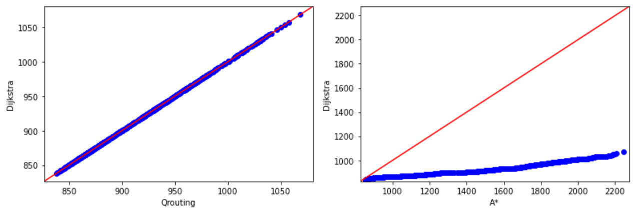

Figure 20. OD=(4,16)

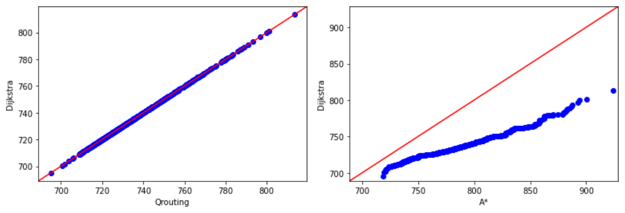

Figure 21. OD=(18,5)

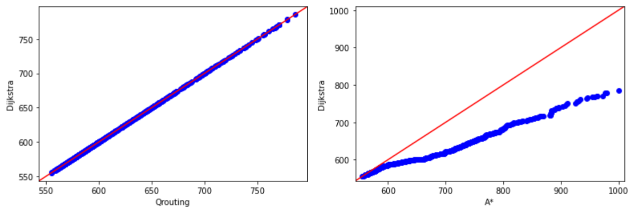

Figure 22. OD=(12,20)

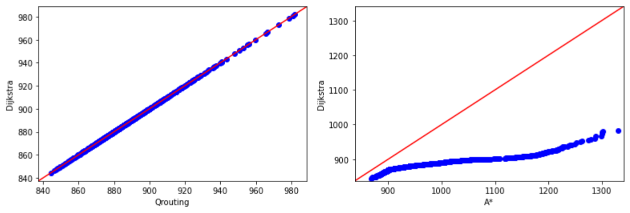

Figure 23. OD=(17,5)

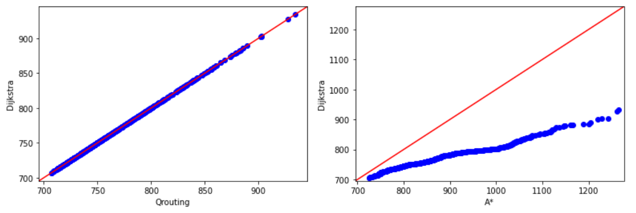

Figure 24. OD=(22,9)

Figure 25. Quantile plot of estimated shortest path travel time for 10 selected OD pairs. All units are shortest path travel time. Graphs have been re-scaled to show 45 degree line

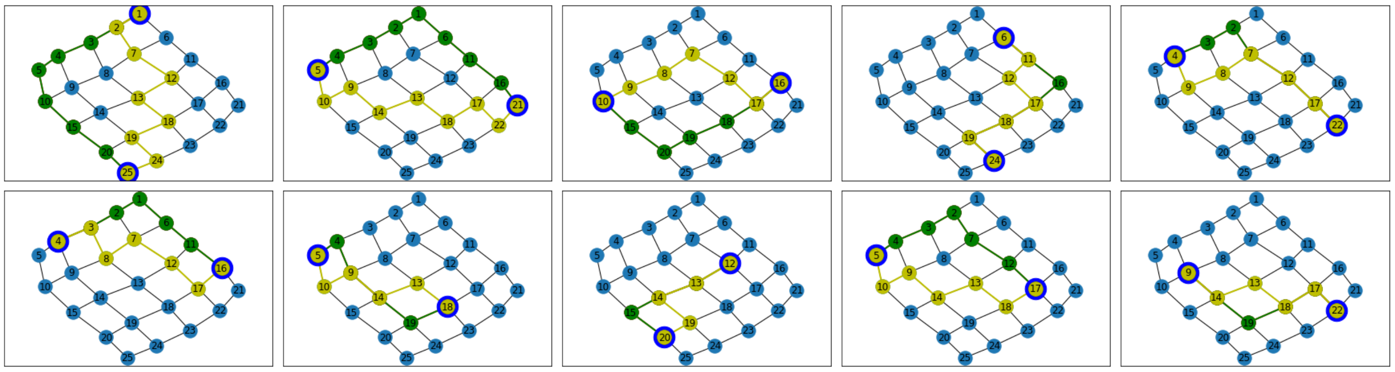

Figure 26. Shortest route after 1000 seconds. OD pairs are in blue. Left to right, top to bottom: OD = (1,25),(5,21),(10,16),(6,24),(22,4),(4,16),(18,5),(12,20),(17,5),(22,9). A* shortest routes are in yellow. Dijkstra shortest routes are in green. Q-routing shortest routes are in red.