CS 229 Final Report: Location Based Adaptive Routing Protocol(LBAR) using Reinforcement Learning

By: Eunjoon Cho and Kevin Wong

Abstract

In this paper we present an algorithm for a location based adaptive routing protocol that uses both geographic routing and reinforcement learning to maximize throughput in our mobile vehicle network. We use reinforcement learning to determine the correct direction to forward a packet and then use geographic routing to forward a packet toward the network sink. We use an extension of the q-routing algorithm, originally proposed by Boyan, et.al [1], except that we apply it on a location based routing protocol that benefits over a link based routing protocol in scenarios where the network is highly mobile.

1. Introduction

Standard wireless routing protocols do not work well in a mobile setting because they were not designed to adapt to a quickly changing environment. The environment we have set is a vehicular sensor network where sensor nodes within each vehicle will communicate with other nodes across the network in a multi-hop fashion.

Routing protocols in wireless sensor networks can be generally categorized into link based and location based routing methods. Link based routing uses the link information between nodes, and can be further divided into proactive and reactive routing. In proactive routing methods, each node keeps information of the network topology and finds the shortest route whenever it needs to transfer a packet. Nodes using reactive routing methods flood the network when needed to send a packet. The sink node determines the optimal route based on its specific algorithm, and then initiates a path between the source and the sink. This path is used until a disconnection is discovered, where the source initiates a flood to search for another path. When the network is highly mobile, like in a vehicular network, topology based methods described above are usually unstable since the link path is not sustained over a substantial period of time. The source repeatedly floods the network to initiate new paths, which increases the overall overhead in the network.

Position based routing methods assumes that each node has geographical location information of itself, its neighbors and the destination. When a message is needed to be sent, each node forwards the packet in a greedy fashion to the location of the final destination. Usually the requirement of a location service for the sensor nodes is an expensive assumption for wireless sensor networks. However, in our vehicular network scenario with the advent of GPS devices we can assume that this is possible.

Another major drawback of using a position based routing method is that there is no link quality measure of the current path that is being used. Since each node has location information of its neighbors only, it is likely that our greedy packet forwarding will lead us to a local minimum. The contribution of our work is that through reinforcement learning we can find routes that may not necessarily optimal in a local greedy sense, but are more likely to eventually deliver the packet to our final destination.

We can imagine a scenario where we would like to send a packet from here to somewhere in San Francisco. When sent via a greedy manner, it is likely that packets will be forwarded to cars on the El Camino real, since this each hop along this path will make it closer to our final destination. However, we know that cars on the 101 or 280 are more densely and evenly distributed. Therefore, even though sending the packet through cars on University Avenue might temporarily get us further from our final destination, we would like the network to learn this path since it will lead us to a stronger connected path and therefore result in a higher throughput.

2. Related Work

Q-Routing

Q-routing was one of the earliest works that dealt with combining machine learning with network routing. Routing algorithms typically route along the shortest path from sink to source in an attempt to minimize hop count and latency. But if the network contained a bottleneck, a link could become oversubscribed and packet losses would occur due to queue drops and packet collisions at the bottleneck. Q-routing attempted to deal with this issue by selecting its next hop among its neighbors that had a minimum delivery times to the network sink. This delivery time included any queuing and transmission time a packet encountered during any transmission. Q-routing defines an equation $Q_x(d, y)$

which defines the estimated delivery time of a node x if it routes through another node y. After forwarding to y, the minimum delivery time to the destination from y is returned to x. This minimum delivery time is defined below in Equation 1, and is used to update node $x's$ delivery time in Equation 2. The values of q and s are the queuing and transmissions times of the packet respectively.

$$t = \min_{z \in neighbor\ of\ y} Q_y(d, z) \tag{1}$$

$$\Delta Q_x(d, y) = \eta \left(\overbrace{q + s + t}^{new\ estimate} - \overbrace{Q_x(d, y)}^{old\ estimate} \right) \tag{2}$$

As a result of using delivery time as a metric to decide routing paths instead of a simple shortest route, the authors demonstrated that Q-routing outperformed shortest path routing in a irregular network during periods of high load. Unfortunately, the gains of Q-routing are only available after the learning algorithm's policies and delivery times have converted according to the above equations. [1]

AntNet

Learning methods have been adapted to Wireless Sensor Networks (WSN) with a more practical view using concepts of ants and stigmergy in AntNet [2]. In this model, forward ants are periodically sent throughout the network in a random manner to the destination node. While an ant moves through the network it keeps track of the node and time it passes through. The destination node selects the path that is optimal in time and then initiates a backward ant back to the source using the same path saved in the forward ant. As the backward ant moves back it updates the "goodness" of the nodes in its path, i.e., increases the probability of that node to be chosen when sending to the same destination. Packets that are later forwarded across the network rely on this probability to select the next best hop to its destination.

It has been shown that various versions of the AntNet algorithm shows good performance in learning the dynamic topology of a WSN. However, given that this is a link based routing method, there is reason to believe that convergence will be difficult in our highly mobile vehicular scenario.

3. Adaptive Location Based Routing

$$\Delta Q(s_t, a_t) = \eta_t(s_t, a_t)\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{3}$$

For our routing algorithm we generally follow the q-learning algorithm given in Eq. 3. We will describe what we have associated with each of the terms in the q-learning algorithm.

State-action definition

In order to simplify potentially complicated road networks for our simulation, we decided to partition our simulation space into regions defined by a rectilinear grid. Each grid block was considered a state and vehicles within that region were associated with that state. It was assumed that the grid boundaries and spacing was global knowledge and all vehicles would be able to unambiguously determine their proper state. Figure 1 has an example of such a grid partitioning of a simulation space.

Using this simplified state space, we define our state actions as forward a packet to a node in a neighboring state. We considered an 8 connected neighborhood for each state, so nodes had potentially 8 possible routing actions at each state. Since each action is associated with sending the packet to a next state, we express action as simply the next state that the action would lead us to, as expressed in the q-routing algorithm. We decided to forward the packet to a node in the next state that was closest to the destination in terms of Euclidean distance.

Reward function

$$SLE(X, Y) = \frac{1}{\sum_{x \in X, y \in Y} PRR(x, y)} \tag{4}$$

Our reward function, which is an estimated measure of link quality and potential throughput from a specific state to another, was defined as a function of the Packet Reception Rate (PRR), which is defined between nodes. PRR is a measure used in routing that measures the probability that any given packet transmission will be successfully received. It can be estimated through active beaconing or previous packet forwarding trials. Generally PRR can change significantly over time due to the inherent variability associated with ad-hoc wireless communication.

We define our State Link Estimator (SLE) reward function, shown in Eq. 4 as an extension to ETX, except it measures the link quality between two states by taking the inverse of the PRR of all routable links between two states. In Eq. 4, x and y refers to a node in state X and Y respectively. ETX is a bidirectional path metric used in WSN that measures the expected number of transmissions required to successfully send a packet between two nodes. It is typically calculated through beaconing or previous packet forwarding trials. Our metric extends ETX to account for the multiple nodes in each state and is proportional to both the link quality as measured through PRR and the number of nodes in both states, which is rough substitute for the possible throughput. Since our calculations only consider unidirectional link estimates, our ETX measurements across states will be somewhat skewed, however, for the sake of simplicity, we are ignoring the effect asymmetric links for our simulation. [3]

Learning Q Values

$$\Delta Q(X,D) = \eta(SLE(X,Y_{min}) + \gamma \min_{Y\in N} Q(Y,D) - Q(X,D)) \quad (5)$$

We used Q-learning to make incremental updates to our Q values for each of our states at regular intervals as shown in Eq. 6, where N is the set of all neighboring states of X, D is the state containing the destination, and Y represents a neighboring state. We chose the learning rate η and the discount factor γ so that our Q values would be stable for our simulation.

Routing Algorithm

$$NextState\,(X) = argmin_{Y\in N} Q\,(Y,D) \quad (6)$$

To make a routing decision at each state, shown in Eq. 6, we find a state in our 8-connected neighborhood that has a minimum Q-value and route a packet toward the node in that state that has a minimum Euclidean distance to the packet's destination. We used GPSR as the underlying routing mechanism when routing between states.

4. Simulation and Evaluations

Evaluation on mobile networks is a difficult problem in that most of the current used simulators for wireless sensor networks are inefficient to scale and mobility. JiST/SWANS is a simulator that has focused on solving these problems and is designed to overcome such problems in simulating mobile communication environments. An extension of this JiST/SWANS simulator also provides strong visualization for understanding the communication process in the mobile network.

Using this simulation tool we have implemented a setting with the actual roads of a small section in Chicago, as shown in Fig.1 We have then placed 100 routable nodes(or cars) on top of these roads with varying speeds similar to the actual movement of a car. Actual communication is limited to one source node at the bottom of the map, and one destination node located at the top left corner of the map. We have set two possible paths for routing. The first route which is sending packets directly north is greedy, but has a weak connection. The other route which is a detour and might be less intuitive to consider has an actually stronger connection, and it is this route that we want the network to

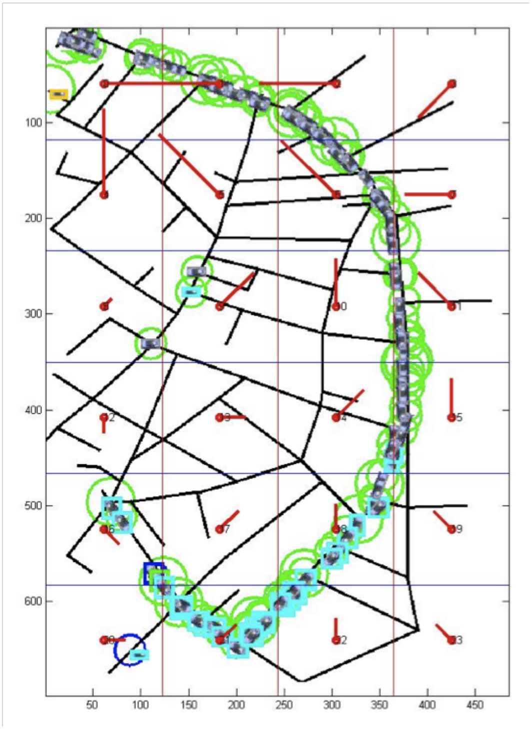

Figure 1: Map of converged decision flow.

learn. We have divided up our map into 4x6 grids, and each block within our grid represents our states.

We have a simple communication setting where the source sends packets to the destination every second, and the simulation lasts for 500 to 1000 simulation seconds. The underlying routing protocol that we use is GPSR, and our LBAR learning algorithm acts simply decides which node would be our next hop over the greedy decision made by GPSR. The destination and source nodes our fixed at static locations whereas all other nodes our dynamic within their path. We use beacon intervals of 1 second for nodes to maintain information of their neighbors and to update the Q values.

5. Results

Route decision convergence

We check whether the network converges to a Q value that gives the correct decision for routing at each state. The convergence of the decisions occur faster than the actual convergence of the Q values. This is because of the fact that decisions are made based on the minimum Q value among its neighbor nodes. A certain state can have a smaller Q value compared to the other states and can be continued to be this way even though there are minimal changes in the actual Q values.

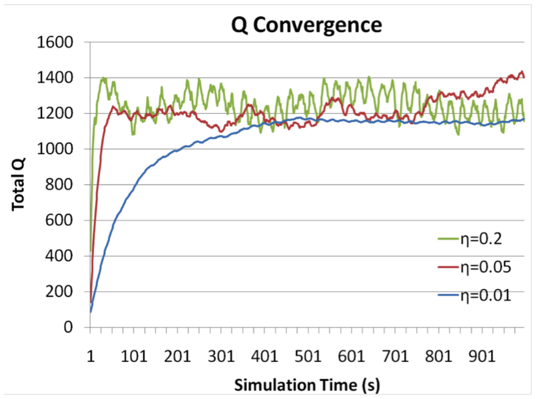

Figure 2: Convergence graph of Q values over different learning rates

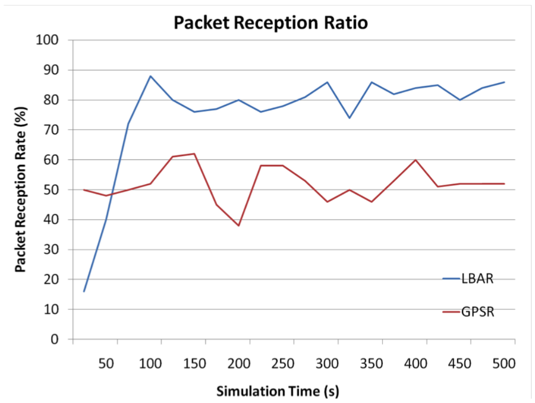

Figure 3: Comparison of PRR at destination to GPSR

In Fig. 1 we can see the final converged decision flow of our simulation setting. The direction of the red arrows corresponds to the next state the network learns to forward the packet. The length of the arrow corresponds to the inverse of the Q values defined at the next hop state, meaning that the larger the arrow the closer, in connection, our next state is to our destination. We can see, for example, that nodes in state 12 learn it is better to forward its packet to state 16 even though it is actually moving further away from the final destination.

Q value convergence

We next check whether the actual Q values of our network show some convergence. We are not guaranteed any strong convergence in our Q table since the reward value(SLE) that we are using is dynamic and can change over time. We have tested that with a fixed SLE that our Q values do indeed converge to a certain value. Although our reward value(SLE) alters over time, we assume that the general distribution over the roads stay somewhat consistent over a period of time. With this assumption we can say that although the links between two nodes may be constantly changing, the links between two states, i.e., the SLE value would be somewhat consistent.

Fig. 2 show the convergence of our Q values over different learning parameters. We can see that with a small learning factor our network is slow in learning but maintains to be stable over a long period of time. On the other hand, when a large learning rate is used we can see that it reaches to a value equivalent to the converged value in a short amount of time, but is unstable. We can adjust our learning factor according to our road environment. If we believe that our road traffic changes rapidly over time during the day, we seek to set a high learning rate so that the network quickly adapts to the new density measure in the roads. If we have a road where the traffic is fairly consistent we can set the learning rate low, so the Q values are stable and the nodes are consistently led to make the correct decisions.

Comparison with a greedy algorithm

We show our final evaluation by comparing the packet reception ratio at our destination with a greedy algorithm. As expected our results in Fig. 3 show that a greedy algorithm can easily face situations where it will not be successful. However, using our adaptive learning algorithm, the network learns over time to find the optimal path to the destination. We can see in our graph that LBAR initially starts of with a lower PRR since it lacks information of which paths are stronger than other. However, over an amount of time that depends on the learning factor, the PRR increases and maintains this link quality until a significant change in the network occurs.

6. Future Work

The LBAR algorithm shows promise in its ability to utilized high throughput paths long road networks in routing compared to greedy geographic routing algorithms. Further refinement of our algorithm could further increase its performance and improve its potential reliability in more complicated traffic patterns.

State Definition Improvements

Our grid-based state space partitioning was efficient in terms of storage complexity and determining neighboring states, but it does not conform well to the underlying road network. Using fixed grid sizes would cause us to either waste state on areas with no roads, or group complicated road sections into the same state, both of which were undesirable. Using a finer grid would help model complicated regions, but would be inefficient for areas with sparse straight roads, as we would have to maintain Q-value storage for states devoid of any roads. Additionally, the grid boundaries would be somewhat arbitrary with respect to the roads, as a state boundary could lie exactly along a road, which could make routing ambiguous or could form routing loops.

One possible alternative is to treat each intersection as a state, and associated mobile users with the closest intersection. Modeling our simulation space in this way would cause the number of states to be proportional the complexity of our road network. We could then define our state actions as forwarding packets to one of the neighboring states, or in our case intersection. This would add extra complexity to our existing algorithm, as we would have to have variable sized data structures to keep track of neighboring intersections and more complicated preprocessing of our road network would be necessary to extract all possible intersections. Additional preprocessing would be also be necessary to ignore short road segments, such as driveways and service roads that would appear in maps, but are not typically used in vehicle through traffic.

Reward Function Improvements

Our current reward function is a simplified version of ETX that has been extended so that it can be defined across states instead of nodes. We originally chose this metric so that we could have a metric that measured both link quality and throughput, so we could identify high traffic roads. To extend our reward function, we could use bi-directional link quality estimates to better characterize individual links. We also do not distinguish endpoints of each link only the total number of links that cross between two states. So if there were a scenario where state A had 10 nodes, and state B had 10 nodes, and if each node in A had a single link to a unique node in B, these ten links might have the same SLE compared to a scenario where state B only had a single node, with which all nodes in state A had shared a link. If the beaconing was infrequently and non-overlapping, each of these ten links could have the same PRR, but the throughput between these states would be very different, since the first scenario could potentially handle ten times the traffic as the second. These two scenarios are not well represented by their respective SLE values. More research into alternative throughput sensitive reward functions between states is clearly warranted.

Larger Scale Testing

We only tested with a modestly sized road map due to time and processing power constraints. A future larger scale simulation could help us identify any degenerate routing cases with LBAR.

7. Conclusion

Our LBAR algorithm shows improvement over GPSR in terms of packet reception rate in our simulated test case. LBAR delivered 83.4% of packets with an $\eta = .01$, while GPSR only delivered 51.4% of packets. The long term packet delivery ratio would be somewhat higher for LBAR, as we counted packet transmissions during the initial route setup and convergence period for our packet reception statistics. This period had frequent packet drops as nodes did not have a reliable nest state to use in packet forwarding. Our simulation environment was setup to specifically show a situation where LBAR should show improvement, but in general, LBAR will favor routing along roads with the highest estimate throughput and link quality. On the other hand, the GPSR algorithm ignores roads and will route toward any node that will get a packet closer to the destination. As in our simulated test case, ignoring roads and traffic patterns completely can lead to suboptimal routes and packet drops do to the inability of nodes to find a next hop.

References

[1] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems 6*, pages 671–678. Morgan Kaufmann, 1994.

[2] G. D. Caro and M. Dorigo. Antnet: A mobile agents approach to adaptive routing. Technical report, 1997.

[3] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146, New York, NY, USA, 2003. ACM.