Data Preservation in a Sensor Network

Bin Tang, Jennifer Ly
California State University Dominguez Hills, Carson, CA 90747
Keywords: Reinforcement Learning, Networks, Data Preservation, Q-Learning, Graphs

In a base station-less sensor network, there can be sensor nodes with an overflow of data packets. In this scenario, the main goal would be to preserve the data packets and not lose the overflow data. Therefore these sensor nodes that need to offload the extra data packets (now referred to data nodes) onto sensor nodes that can take on the overflow (storage nodes). But it takes energy to send the overflow to the storage nodes(communication costs); and the data nodes' main goal is to preserve the data. Therefore to address both constraints, we need to find the most optimal path assignment to find where a particular data node can offload its data to a storage node; while incurring the lowest possible cost of offloading the data from a data node to a storage node.

With this goal in mind, we look to reinforcement learning to potentially find this optimal solution. The basic idea is to have an agent interact in a given environment and get evaluated for how good or bad its actions are based on its current state. We would want to design an agent that is from a set of data nodes that will try to find the best spare storage node to offload their data to. Tham and Renaud also proposed promising aspects in involving multiple agents to interact within the same environment. They were able to find the most 'balanced' approach based on information given about the entire environment and how each of the agents were doing.

In this proposal, we are start with a simple sensor network that is composed of 4 nodes. Two are data nodes and the other two are available storage nodes. They are placed in an alternating manner (eg. storage, data, storage, data). From this example, we can see that the best solution is to have each data nodes offload to the storage node on their left. Both data nodes need only to travel a total distance of 1 hop in order to offload their data. If we consider an alternative solution, the remaining data node will need to travel 3 hops in order to offload that data packet. This is where we consider reinforcement learning to train the data nodes to find the best storage node to offload their data to. Here, we design and implement a common algorithm (Q-Learning) to help find the best paths for all data nodes. We want to add factors that will determine the decision quality of each data to storage node assignment. Such considerations would be, the type of node and the number of hops(distance) from a data node to a storage node. We expect that this implementation will provide a generalized solution that can also address larger sensor networks(eg: a KxK grid), and achieve our two primary goals: data preservation and lowest possible communication costs.

The implementation of our experiment required the following to be known: the environment(sensor network), the data nodes, and available storage nodes. The program first sets up our K-Sensor network where the nodes are labeled 0 to K^2-1. There, we assign the data nodes and storage nodes accordingly. For the reinforcement learning portion, the last prompt is to indicate how many episodes the agent will run. Once that is provided, the agent will randomly select a data and storage node and find the shortest path. Our Q-Learning algorithm will evaluate the quality of this assignment and update it accordingly. The Q Table will be returned at the end and added alongside all the others from the remaining episodes. This is a way to mimic cooperative learning in the sense it does take past experiences into account. Therefore, once the last episode ends, we are left with a cumulative Q-Table that will highlight the most balanced approach to assigning the storage nodes from a given data nodes. If there was competition for a storage node, then additional rankings would need to be done. From the Q Table, the

difference between each data nodes' maximum and minimum values are taken to evaluate the value impact each node has. Then the storage node is chosen linearly based on the data nodes ranking. Once the assignments are done, we take the sum of their cost values. Afterwards, the data node list, storage node list, and Q Table will get forwarded over to create a directed graph to run the Minimum Cost Flow(MCF) algorithm for comparison. The Q Table values will be negated, since MCF looks for lowest cost., and the Q Table is reflected the highest value of each action. The results would be optimal if the sum cost value is equal to the cost returned by MCF.

From our simulations from the basic 4 node base case to a 'K' sensor grid network, the assignments from our algorithm was comparable to the Minimum Cost Flow algorithm. The simulation ran for 25 trials where in each trial, 3 data nodes and 3 storage nodes would be randomly assigned. Within each trial will have the agent run for 100 episodes where the agent will randomly explore random data to storage node assignments. In 22 trials the result was comparable to the MCF algorithm, the returned costs were equal to each other. The other 3 trials were disqualified as the agent got stuck during the exploring phase. This can most likely be remedied by adjusting the learning parameters, but as of now it needs to be manually done.

Overall, the use of Reinforcement learning techniques seems promising since it went from theory to implementation. Along with the fact that, it received comparable results to MCF. There is further improvements to be made in regards to: tuning the parameters automatically instead of manually, different capacities for storage nodes, and using available machine learning libraries to make the experiments more scalable and robust.

References

Tham, C.-K., & Renaud, J.-C. (2005). Multi-Agent systems on Sensor Networks: A Distributed reinforcement learning approach. *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. https://doi.org/10.1109/issnip.2005.1595616