Belal Wang

Michelle Beryl



1. *MOTIVATION:*

*“Recently wireless sensor network (WSN) has been widely used for monitoring railway tracks and rail tunnels. The key requirement in the design of such WSN is to minimize the energy consumption so as to maximize the network lifetime. ”[1]*

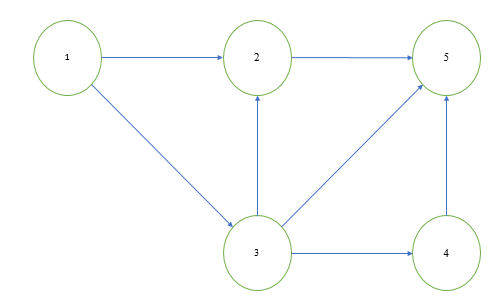
This project uses the concepts of Constrained Optimization and Network Utility Maximization from class and applies them to the idea of maximizing network lifetime. This proves to be interesting to study because it is useful in real-world networks and has a wide range of applications. In cases such as wireless sensor networks, an important factor is the maximization of the network lifetime, since it is not always practical to replace or recharge the sensors (nodes) and we want the network to be fully operational for as long as possible. [1] In exploring the concept of network and node lifetimes, we brainstormed a few ideas:

* Opportunistic routing and calculation of expected travel time from source to destination
* Shortest path as a priority metric for opportunistic routing; lifetime as a shortest path tiebreaker
* Ratio of successful packets received VS total packets sent

In the end, we selected a problem related to maximization of a networks lifetime, measured in the number of packets to be sent before failure.

1. *PROBLEM FORMULATION:*

Consider the network shown in figure 1 below.





Our aim is to Find the maximum number of packets that can be sent (network lifetime) on a given network from a source node (1) to a destination node (5), such that:

* Packets are sent one by one along some path Pi, each path containing some set of nodes n ε N, N={1,2,3,4,5}
* Each node n has some lifetime Ln and some energy cost En
* Lifetime decreases by En every time the node is used
* When a node reaches a lifetime <0, that node is removed from the network, and all paths that use the node are invalid.

This involves finding the combination of paths that maximizes the network lifetime. When the lifetime of a node ends, it is removed from the network and the packet needs to find a different path from the source to the destination. The network ‘dies’ when nodes are removed to the point no paths exist from the source to destination.

*C. MODELING THE PROBLEM :*

We approached this problem using the concept of network utility maximization (but in this case, it’s an integer program instead of a convex program), where we want to maximize the total number of packets sent on each path i before the lifetimes of nodes in that path end. We maximize this subject to the constraints that the energy consumed by these packets does not exceed the lifetime of each node. Once the lifetime is 0, that node is removed from the network and the remaining nodes are considered. This is described by the following Integer Program:

Maximize: 

Such That: 

Where:

   
 P is the set of all paths  
 is the total packets sent on path *i*  
 is the maximum lifetime of node n  
 is the energy cost per packet on node n

All variables are positive

Furthermore, we made the following assumptions in order to find a solution:

* We assume all distances between the nodes here are 1 unit each.
* In our case, we also make an assumption that all energy consumption values for intermediate nodes are 1 unit each and the lifetime of each intermediate node is 5.
* Lifetimes of the source and destination nodes are infinity.

*D. Problem Solving Methods*

The original idea was to use **Brute Force/Permutation**. For small networks like Figure 2 below, paths could be enumerated and the solution can be found by permutation and some intuition. For larger and more complicated networks however, this became much more difficult to solve, requiring PLn time to complete.

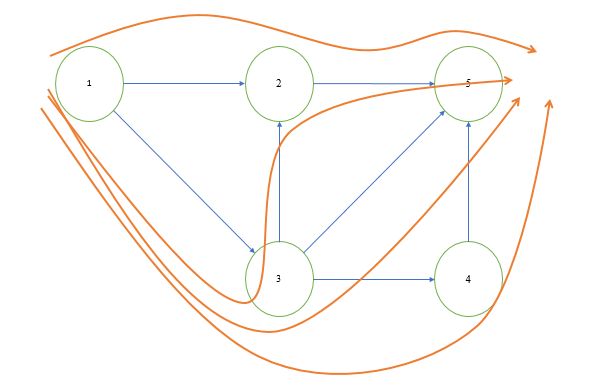


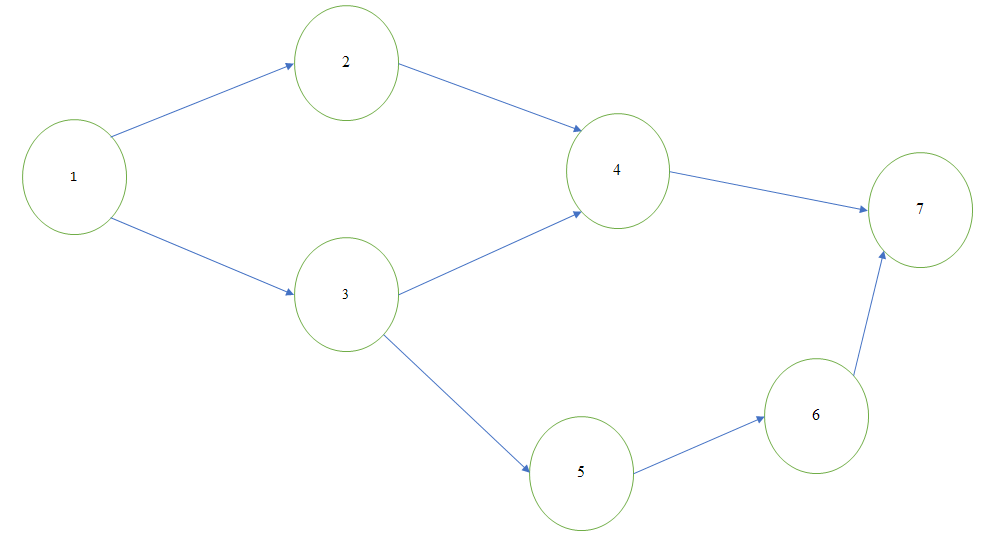
Figure 2: Network with paths from nodes 1->5

Next, we thought of ways to decrease the valid permutations used by the brute force solution. This would allow for larger, more general networks to be analyzed. The intuitive next step was considering the set of paths that maximizes the number of packets sent through the network as the set of paths that uses the **shortest paths from source to destination**. We modeled the problem such that the paths with minimum number of nodes from source to destination were selected first. Below is the thinking and framework for a proof:

*When picking a path, the 'shortest path' will contain some set of nodes in N, number of nodes equal to the number of hops of the shortest path 'h'.*

*For h+1: the set of nodes in a given path will either be disjoint from the previous hops, or contain some overlap. In the case of disjoint, the new paths do not affect the validity of lower hop shortest paths. In case there is an overlap, the lower hop shortest path is preferred as the total number of nodes used is lower (and thus total network lifetime is extended). There will be some 'bottleneck' node with lowest remaining lifetime that is shared between the h and h+1 paths, and using the h shortest path allows this node to die while maximizing lifetime of other nodes.*

* For the network in Fig.1, the shortest path chosen first will be either 125 or 135. If 125 is chosen first, then we can send 5 packets before the lifetime of node 2 expires. Then we can send 5 more packets through 135 until the lifetime of node 3 expires. Thus, a maximum of 10 packets can be sent through this network before the lifetime ends. Node 4 is not used in this case, but it doesn’t matter since the number of packets sent are optimally maximized before the network lifetime expires.
* Another general situation where the shortest path approach will work is if we have a network of disjoint paths, or if the overlapping paths don’t include a shortest path.
* However, this approach worked for some networks but not all, and we were able to find a counterexample for this in Fig.2 shown below with the same assumptions as our previous example and node 1 as the source and node 7 as the destination.





In this counterexample, there are 2 possible shortest paths, 1247 and 1347. If 1247 is chosen first, then 13567 will be chosen next and the network lifetime will be maximized. But, if 1347 is chosen first, we would not be optimally maximizing the number of packets that can be sent, since the lifetimes of 3 and 4 will end and both paths 1247 and 13567 cannot be used.

* If we modified it with a tie-breaker a stated below, we would be able to solve this example with the modified shortest path approach.

Tie-breaker : If there is a tie for the shortest path, choose the first path to be the one that doesn’t contain a node that leads to an alternate path to get from source to destination. In our example, it would mean choosing path 1247 since path 1347 contains node 3 that has 2 ways of getting to the destination, so 13567 would be an alternate way for path 1347 to get to the destination.

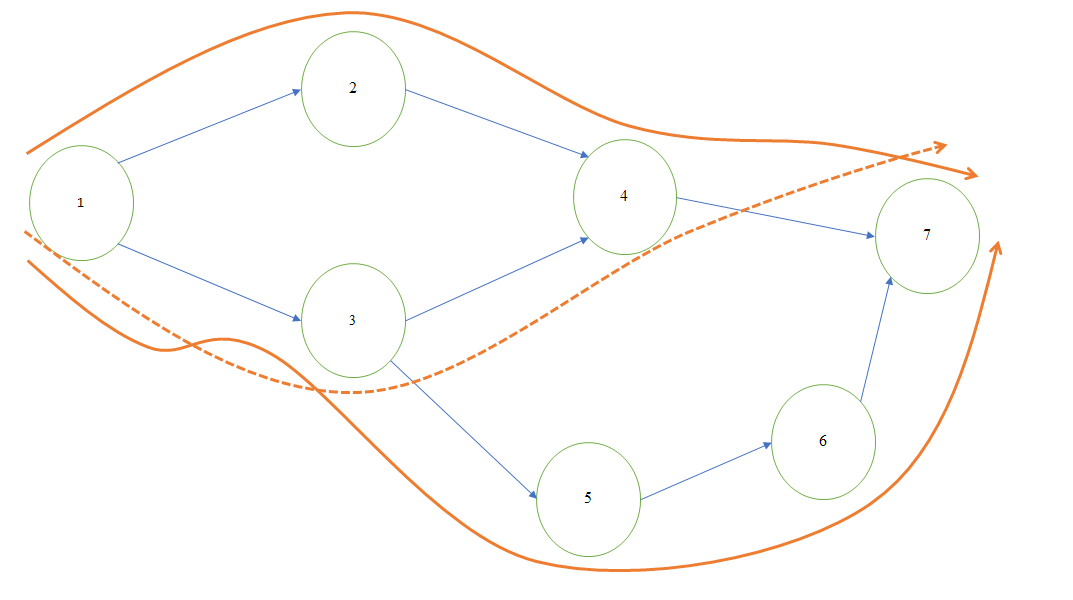


Figure 6: Counterexample Network with viable paths in solid arrows

Thus, when 2 or more shortest paths share a common node, the path that has NO alternate way to get to the destination (through one of the nodes in it) is chosen.

* This leads us to the concept of “Pruning”. If a shortest path is a ‘subset’ of another path ‘C’ from the source to the destination, and there is no other path using this path C (or one of it’s nodes) to get to the destination, then the nodes in path C that do not belong to the shortest path can be removed (do not have to be considered). To explain this, consider example 1. Shortest path 135 is a ‘subset’ of the path 1345 and hence node 4 does not have to be considered.

In other words, if a path is a superset of a shortest path, the additional nodes in this path can be removed.

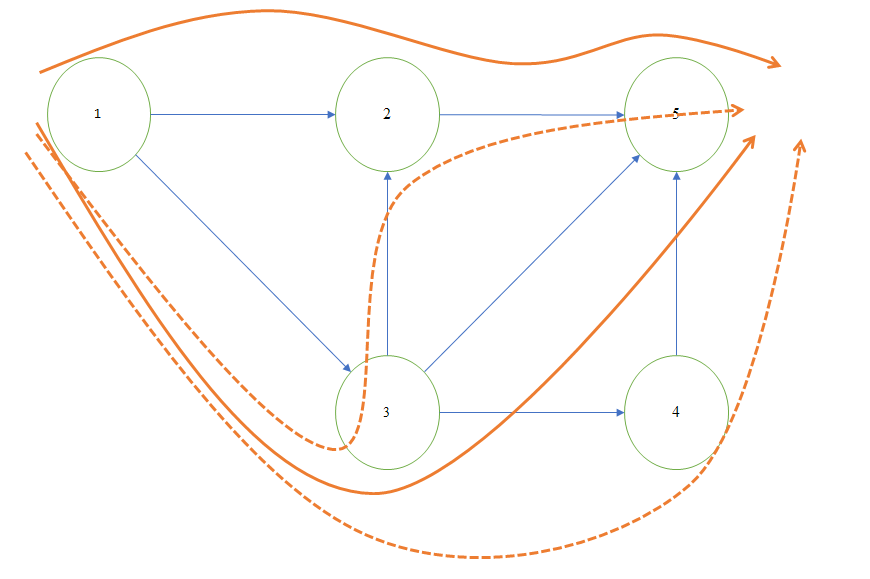


Figure 3: ‘Pruned’ paths with viable paths in solid arrows

Our last approach was to **use a computer program** to choose the order of paths randomly and run that program for a large number. In each run we would simulate a possible combination of paths, and as number of runs increase, we should get a value close to the maximum number of packets that can be sent through the network (maxP), given the lifetime of all nodes. As the number of runs is increased, in our case to 200, we get closer to the optimal solution.

*E. CODE*

See Attached Files

*F. CODE RESULTS*

The code we wrote outputs an approximation of the maximum and minimum packets sent. As the number of simulations increases, we can expect that the approximation would become close to the real maximum and minimum of the network. These values give some expectation as to what the bounds of our solutions should be, and allow us to analyze other algorithms for correctness.

We could use this and compare to the outputs from the shortest path solver to see which approach gives more accurate results. Applied to different networks, we could also identify shortcomings of the shortest path approach.

*G. SOLUTIONS*

*i) Initial Network*

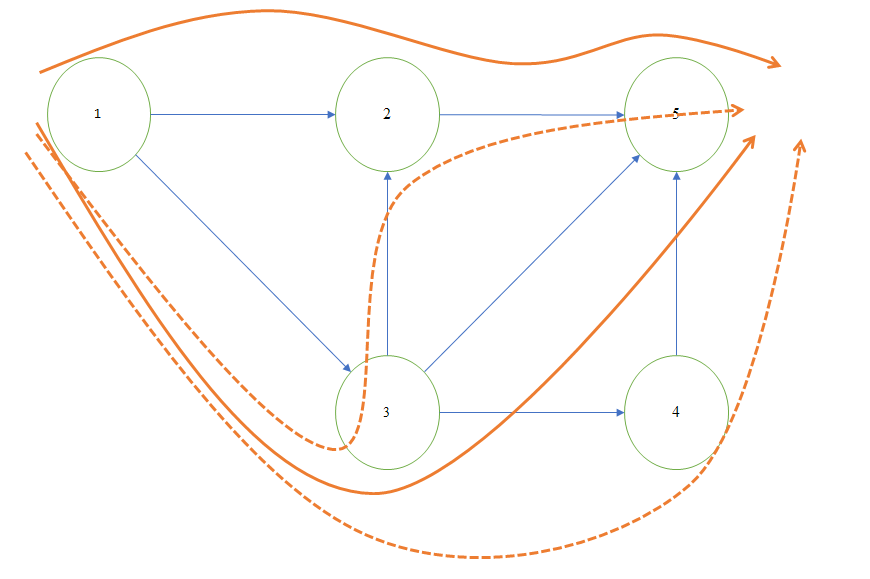


Figure 3: ‘Pruned’ paths with viable paths in solid arrows

* For this network, there are four possible paths:  
  P1={1 2 5}; P2={1 3 5}; P3={1 3 2 5} ; P4={1 3 4 5}
* Because P4 is a superset of P2, it could be removed from the set of paths, leaving P1={1 2 5}; P2={1 3 5}; P3={1 3 2 5}
* P3 could not be removed as it is also a superset of P1. Thus the solution is a combination of P1 and P2 and P3.
* With node lifetimes of 5, we can see that the maximal lifetime is achieved when P1 is used 5 times and P2 is used 5 times, in any order. P3 would not be used as we can see that it would reduce lifetime of P1 as well.
* Our MATLAB program resulted in the same solution:  
  minP=5  
  m**axP=10**pathsmax={P1 P1 P3 P3 P3 P1 P1 P3 P1 P3}

*ii) Counterexample Network*

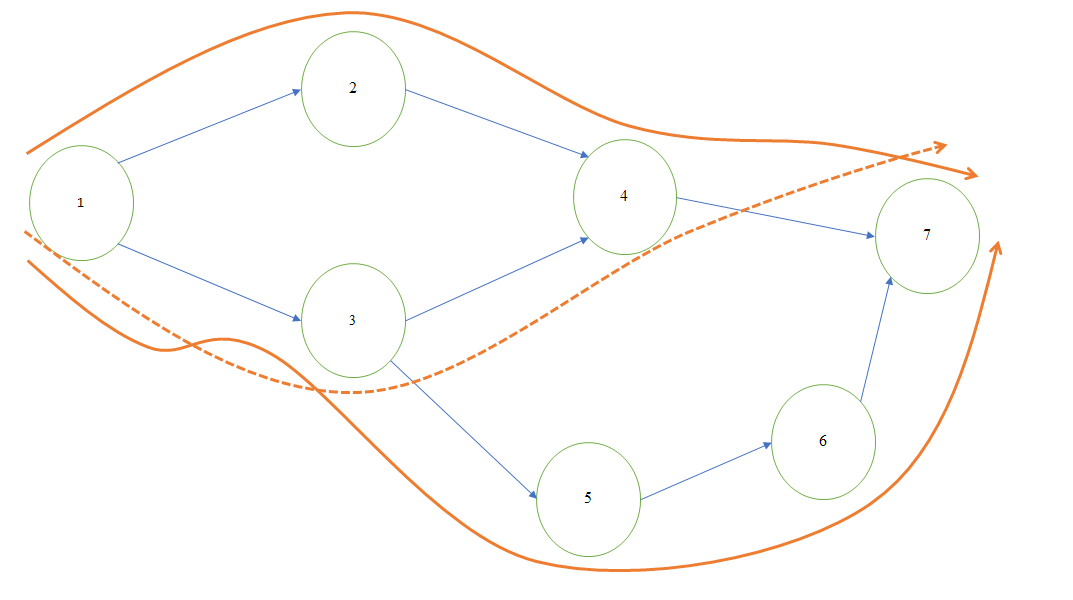
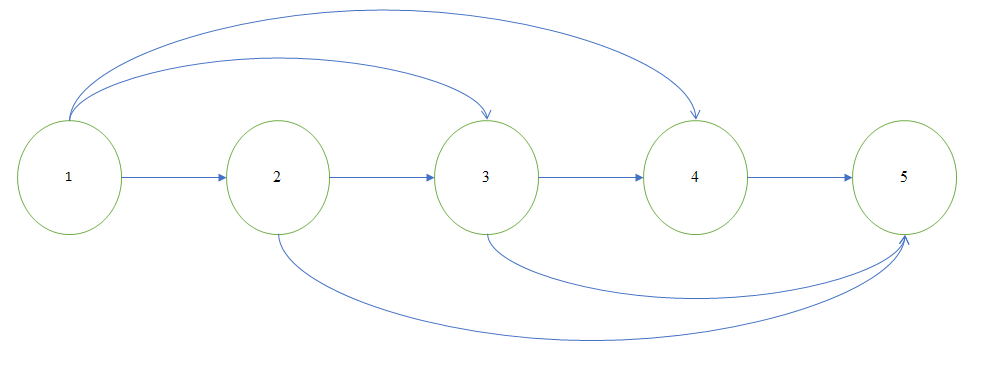


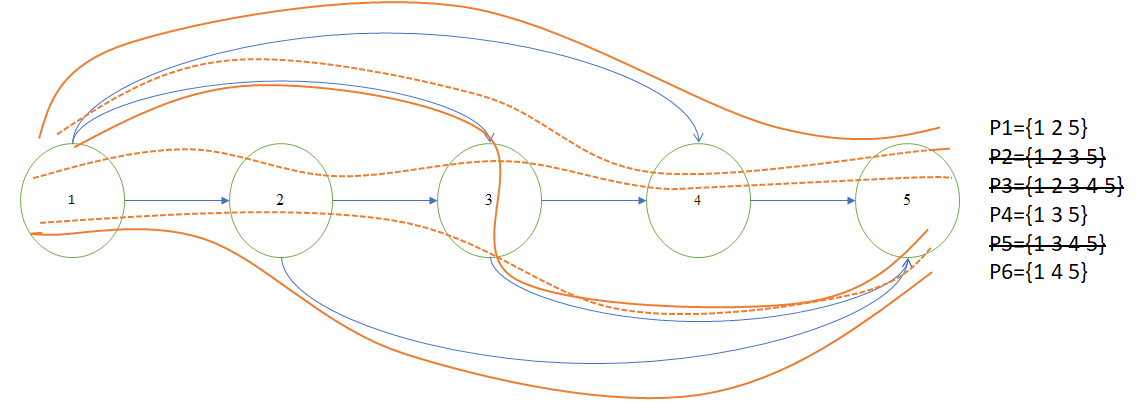
Figure 6: Counterexample Network with viable paths in solid arrows

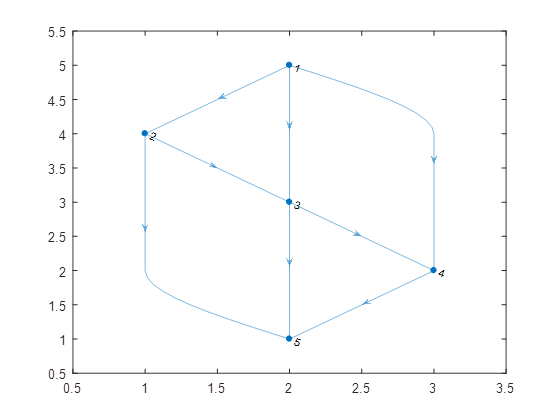
* This network has 3 paths available:  
   P1={1 2 4 7}; P2={1 3 4 7}; P3={1 3 5 6 7}
* Both P1 and P2 can be ‘shortest path’ using ‘# of hops’ or ‘total lifetime drained’ priorities.
* However, using P2 first will not result in an optimal solution, as it uses both nodes 2 and 3 which are required for P1 and P3
* An optimal solution uses only P1 and P3, as these are disjoint, and a maximum of 10 packets can be sent using 5 P1s and 5 P3s

Our MATLAB program resulted in the same solution:  
minP=5  
m**axP=10**pathsmax={P1 P3 P3 P3 P3 P1 P3 P1 P1 P1}

*iii) Other network 1 (using simulation)*





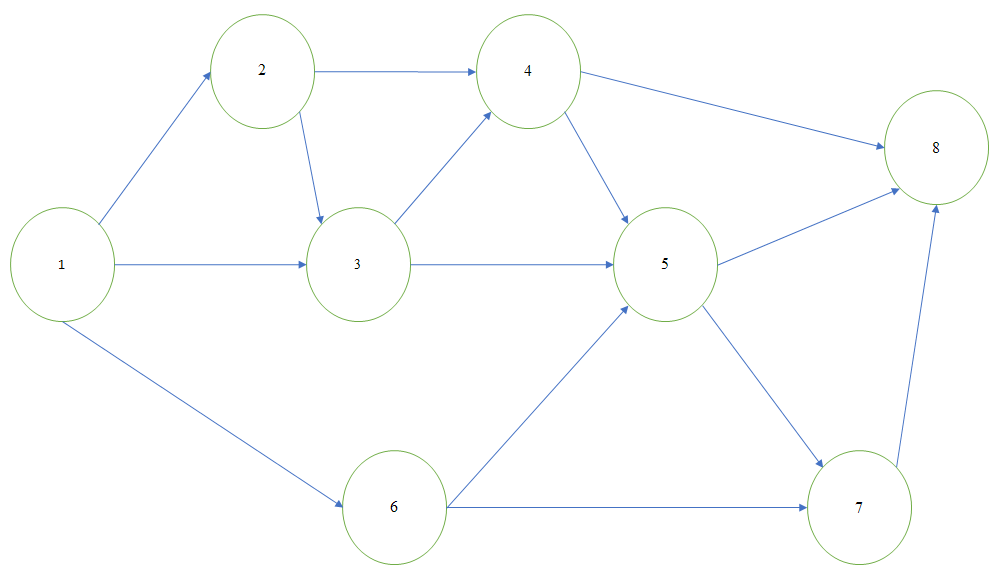


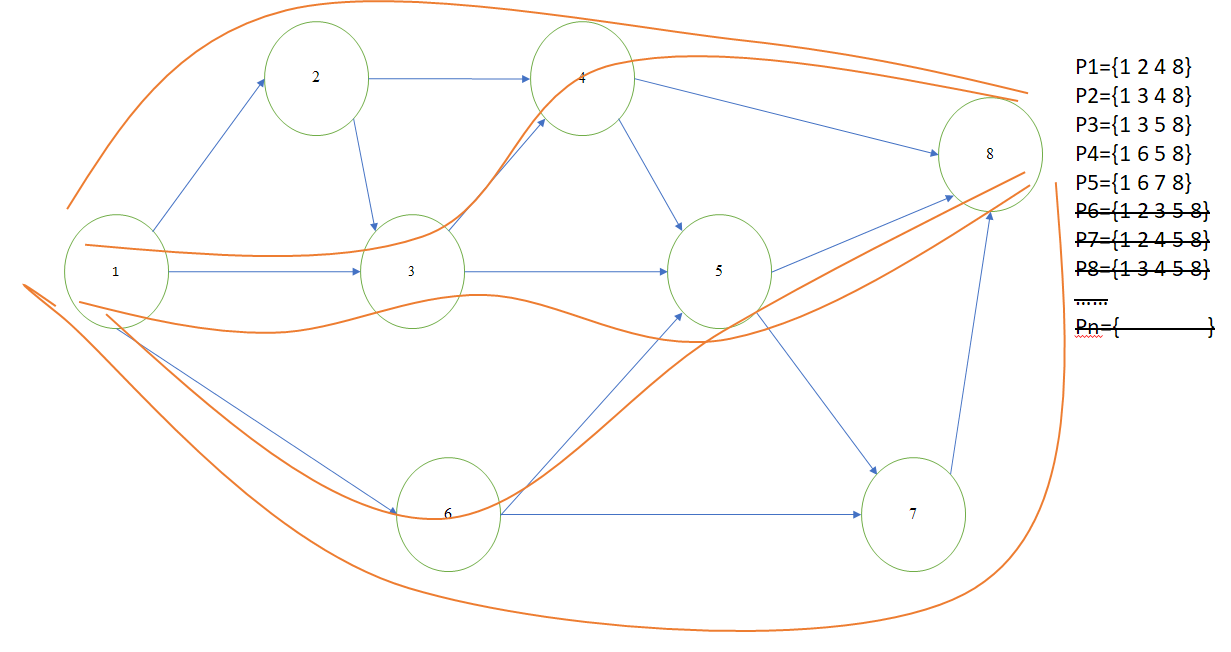
minP=6

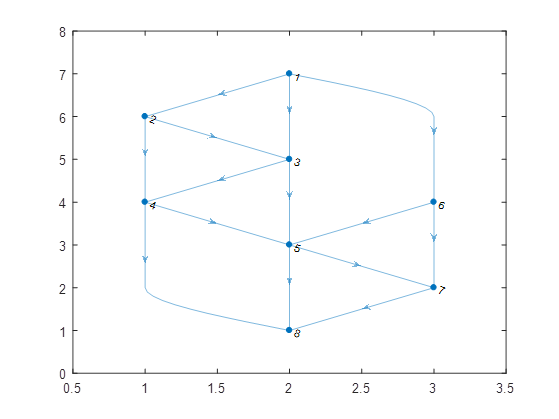
**maxP=14 (close to 15)**

pathsmax={P6 P1 P4 P1 P1 P6 P4 P4 P1 ~~P5~~ P4 P6 P1 P6}

*iii) Other network 2 (using simulation)*







minP=5

**maxP=13 (close to 15)**

pathsmax={P3 P1 P3 P5 ~~Pn~~ P4 ~~Pn Pn~~ P1 P5 P5 P5 P1}

*H. CONCLUSIONS*

* From our first counterexample, we learnt that the shortest path approach we used doesn’t always work. More specifically, it fails in cases where the shortest paths are not disjoint (share a common node).
* But, we could modify it with a tie-breaker like the one stated above, and would be able to solve this example with the modified shortest path approach.
* Since the shortestpath function used in our code does not list the paths that tie for shortest path, we were not able to add this tie-breaker into our code.
* But our shortest path code always picks path 1247 over path 1347, and hence we get the same solution for maximum number of packets while comparing both codes (shortest path and random path) for this particular example.

*I. POSSIBLE EXTENSIONS*

* If we had more time, we could vary the lifetimes of the nodes in the network and maximize the number of packets sent before the network lifetime expires.
  + A possible conclusion we intuitively get from varying the node lifetimes is that node 4 in the example in Fig. 2 is the bottleneck node. If the lifetime of node 4 is smaller than that of the other nodes, the network lifetime is equal to the lifetime of node 4.
* We could also try varying the energy consumption values for each node, to see how the number of packets and network lifetime can be maximized when a different amount of energy is consumed as the packet passes through each node.
* We could try different metrics in our shortest path calculation (number of adjacent nodes, number of intersecting nodes, tiebreakers), that might help develop an analytical solution.
* Instead of randomly selecting paths, ordering them in various combinations.

*J. REFERENCES*

[1] Philipose, A., & A, Rajesh. (2016). “Investigation on energy efficient sensor node placement in railway systems”. *Engineering Science and Technology, an International Journal*.

Found online : <https://www.sciencedirect.com/science/article/pii/S2215098615001743>

[2] “pathbetweennodes.m”, github.com/kakearney; MATLAB code to enumerate all paths given a digraph