Source-Destination Connection in Brain Network

May 27, 2018

Li Kaijian, 515030910566

Contents

1 Introduction

Determination of source-destination connectivity in networks has long been a fundamental problem, where no one has tried solve this problem in a brain network. In this research I introduces the basic information about brain network and give a brain network model. Then I define the source-destination connectivity determination problem formulation on brain network model and prove that this optimization problem is NP problem, which means this problem cannot be solved in polynomial time unless P=NP. I give an optimal algorithm based on MDP. But this algorithm is time computational. I further proposed three approximation algorithms. Finally I use matlab to make a simulation to compare these algorithms.

In section one I will introduce some basic information and features about brain network. Similar content can be find in [1].

1.1 What is Brain Network

The functional organization of the brain is characterized by segregation and integration of information being processed. A central paradigm in modern neuroscience is that anatomical and functional connections between brain regions are organized in a way such that information processing is near optimal. Functional interactions seem to be provided by synchronized activity, both locally and between distant brain regions. Brain networks thus consist of spatially distributed but functionally connected regions that process information. Figure 1 shows a instance of brain network.

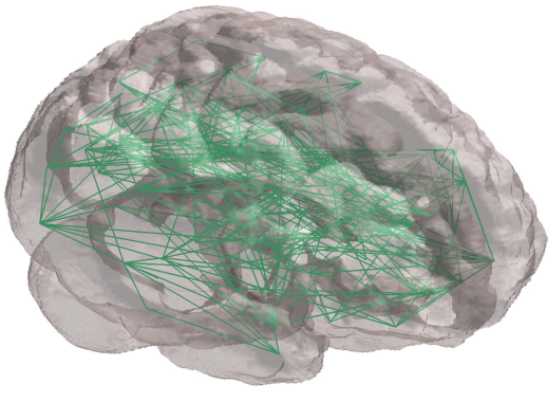

Figure 1: a brain network

1.2 Different Kinds of Brain Network

There are two different but related forms of brain network.

- Anatomical connectivity, also called structural connectivity, which forms the connectome through synaptic contacts between neighboring neurons or fiber tracks connecting neuron pools in spatially distant brain regions
- Functional connectivity which is defined as the temporal dependency of neuronal activation patterns of anatomically separated brain regions.

Figure 2 shows how we can get these brain network from a true brain. More and more studies are focusing on brain network with graph theoretical analysis. In this report I will focus on functional network.

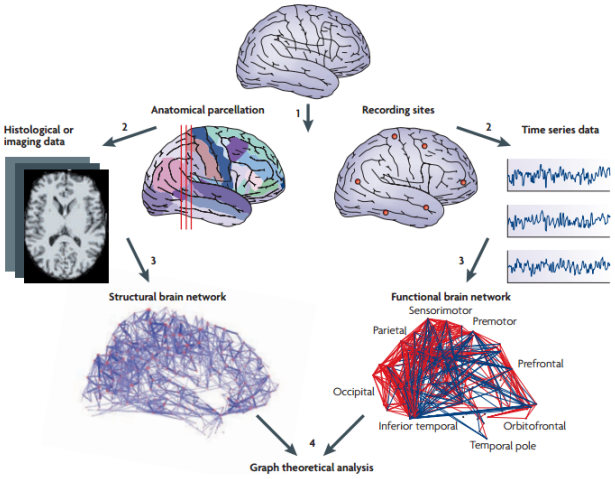

Figure 2: two different kinds fo brain network

1.3 Features of Brain Network

Though it's has been a long time since people start to study brain network, we still don't figure out the pattern of brain network. Graphical models provide means to characterize complex brain connectivity networks, so-called brain graphs. Graphs may be constructed for anatomical networks as well as for functional networks. Thus, they offer a theoretical framework to describe the structural and functional topology of system-wide brain networks. Here I will introduce the some theoretical concepts for Graphical Models and some features of brain network. These features provides a way to construct the brain network model.

1.3.1 ROI and Modules

Considering the functional organization of the brain into local interactions performing low-level information processing, called regions of interest (ROI) or modules, and long-range couplings supporting distributed information processing and providing control and high-level information fusion, brain networks form graphs intermediate between regular graphs where only nearest neighbor nodes are connected and random graphs where all nodes are connected randomly. Functional networks thus form graphs G(V, E), where ROI are called vertices $\{V|vn : n = 1, ..., N\}$, and long-range couplings correspond to edges $\{E|e_{nm} : n, m \in \{1, N\}\}$ indicating key pathways of information processing in the brain. The figure 3 shows this feature.

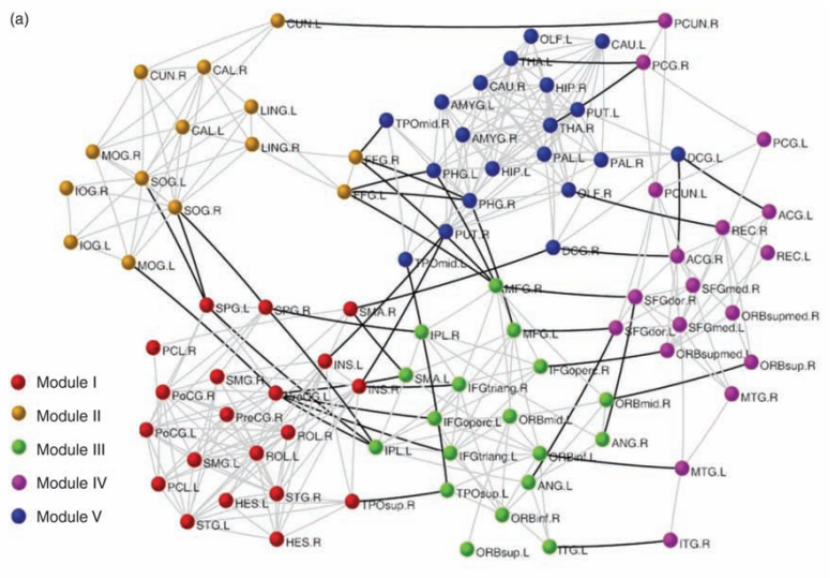

Figure 3: ROI and Modules

1.3.2 Degree Distribution

Degree distribution is a global measure of a graph. $P(V, E|K)$ is the likelihood of a vertex to have degree k. From the researches show that the brain network has a non-Gaussian degree distribution.

1.3.3 Clustering Coefficient and Path Length

local clustering coefficient C(v) measures if all directly connected neighbors w ∈ U(v) of node v are also connected to each other. It's one of the local measures.It is related to the presence of the triangle motif in a network

and represents the local connectivity or cliquiness of the node. An average over all vertices of the network yields the average cluster coefficient C_G which provides a global measure of the network connectivity and represents the likelihood of neighboring short connections.

A path length L is the distance form one node to another node. Average path length is the average length of all path between any two nodes in the brain network. Is represents the probability of a long path and reflects the degree of integration of the given graph.

1.3.4 Small World Properties

If a network has a high C(many short connections) and small L(few long connection), this feature is called small world properties[2].

1.3.5 Dynamic

The brain network is always changing. So it's difficult to use a graph to describe it. Here I will ignore this feature, seem the network is immobile as many studies do.

2 Related Work

2.1 Related Work in Brain Network Model

There isn't a universal graph model for the brain network. All the exiting model just focus on special feature. Here I choose the exponential random graph to build the brain network as in [3].

$$P_\theta(Y = y) = K(\theta)^{-1} exp(\theta^T g(y))$$

Here y is an instance of brain network, g(y) is the feature vector of the brain network y and θ is the weight vector of feature vector. $K(\theta)^{-1}$ is to make sure that the sum of probability is 1.Figure 4 shows all important metrics.The GWD, GWESP, and GWDSP statistics is discussed in [4]. GWD is to control the node degree distribution. GWESP is a local efficiency metric, which is similar to C. GWNSP is a global efficiency metric, which is similar to L. A larger GWNSP means there is a high probability to have a long path. Here I will just give the equations of these metric.

Metric	Description
Edges	Number of edges in network
Two-Path	Number of paths of length 2 in the network
k-Cycle	Number of k-cycles in network
k-Degree	Number of nodes with degree k
Geometrically weighted degree (GWD)	Weighted sum of the counts of each degree (i) weighted by the geometric sequence $(1-\exp\{-\tau\})^i$, where τ is a decay parameter
Geometrically weighted edge-wise shared partner (GWESP)	Weighted sum of the number of connected nodes having exactly i shared partners weighted by the geometric sequence $(1-\exp\{-\tau\})^i$, where τ is a decay parameter
Geometrically weighted non-edge-wise shared partner(GWNSP)	Weighted sum of the number of non-connected nodes having exactly i shared partners weighted by the geometric sequence $(1-\exp\{-\tau\})^i$, where τ is a decay parameter
Geometrically weighted dyad-wise shared partner (GWDSP)	Weighted sum of the number of dyads[a] having exactly i shared partners weighted by the geometric sequence $(1-\exp\{-\tau\})^i$, where τ is a decay parameter
Nodematch	Number of edges (i,j) for which nodal attribute i equals nodal attribute j (e.g., brain location of node i = brain location of node j)

[a]node pair with or without edge

Figure 4: Subset of explanatory network metrics

$$GWD(y,\tau) = (e^{\tau})^2 \sum_{i=1}^{n-1} [(1-e^{-\tau})^i - 1 + ie^{-\tau}] GWD_i(y)$$

$$GWESP(y,\tau) = e^{\tau} \sum_{i=1}^{n-2} [1-(1-e^{-\tau})^i] GWESP_i(y)$$

GWNSP and GWDSP have the same formulation with GWESP.

2.2 Related Work in Source-Destination Connection

There is similar work about random graph[5]. My time complexity proof and the solution algorithms of this problem are inspired by the idea in it.

3 Model and Problem Formulation

3.1 Exponential Random Graph Model

We denote a exponential random graph by G=(V,θ,g,c), where V is the set of vertices, θ is the weight vector in exponential random graph, g is

function to transform the network into a p-dimensional feature vector. c is the cost of edges in G.This work is different with [5], because the existence probability of each edge is dependent and there may be an edge in any two vertices. Let G be a underlying deterministic graph. The probability of G is:

$$P_\theta(G) = K(\theta)^{-1} exp(\theta^T g(G))$$

3.2 Problem Formulation

Here we define three concepts: temporary state, adaptive testing strategy, The Connectivity Determination Problem. These definition can all be find in [5]. But for convenience I will introduce them again here.

- (Temporary State) A temporary state s of a exponential random graph G=(V,θ,g,c) is an $\frac{|V|(|V|-1)}{2}$-dimension vector with element 0, 1 and *. And we define S = $\{0; 1; *\}^{\frac{|V|(|V|-1)}{2}}$ to be the set of temporary states associated with G. Additionally, we denote the condition of edge e in state s as s_e. For a temporary state s, we define it to be a terminating state if either the edge set $\{e|s_e = 1\}$ forms a superset of an s-t path in G or edge set $\{e|s_e = 0\}$ forms a superset of an s-t cut in G. We successfully determine the s-t connectivity by reaching a terminating state.

- (Adaptive Testing Strategy) An adaptive testing strategy is a mapping π : S $\rightarrow$ $E \cup \{\perp\}$. Initially starting from the all- state, an adaptive testing strategy specifies which edge to test (or terminate as denoted by $\perp$) based on the previous testing outcomes.

- (The Connectivity Determination Problem) Given an uncertain directed graph G=(V,θ,g,c) with two nodes s, t $\in$ V designated as source and destination, respectively, the goal is to find an adaptive testing strategy π that incurs the minimum expected cost.

4 Computational Complexity

Here I will investigate the time computational complexity of the connectivity determination problem in brain network.By demonstrating the hardness of two closely related problems, we show both computing the testing strategy with the minimum expected cost holistically and sequentially are NP-hard.

Theorem 1. The decision version of Connectivity Determination Problem is NP-hard

Proof: Inspired by [5], here I use the similar idea to proof it. I prove the theorem by reduction from the s-t reliability problem [6]: Given a directed graph G and two nodes s and t, the s-t reliability is to compute the probability of s being connected to t assuming the edges in G exist independently with probability $\frac{1}{2}$. As s-t reliability problem is #P-hard [6]. I make a little change that let $|E|$ always be $\frac{|V|(|V|-1)}{2}$, obviously this problem is #P-hard.

The reduction performs as follows.For a graph G(V), we transform it to a exponential random graph G(V,θ,g,c) by adding en edge M between s,t and set the rest of G as just the same. The cost to test M is $\frac{|V|(|V|-1)}{2}2^{\frac{|V|(|V|-1)}{2}+1}$ and the cost of testing other edges is 1. Then let $\theta = \vec{0}$. So the probability of all edges is $\frac{1}{2}$.

Now this formulation is just as it in [5]. The following step is just the same in [5].

Theorem 2. Deciding the optimal first edge to test (the edge tested by the optimal strategy in the initial state) is NP-hard.

This proof is also similar in [5]. I use the set cover problem to proof it. [5] has already proofs that this theorem is right for random graph. So if I can use exponential random graph to make the same construction, this theorem is proved. There are two different part. The first part is in exponential random graph any two vertices may have an edge, so the set cover can't be transform into the same graph formulation in [5]. But if we set the cost of these redundant edges as infinite. Then it has the same result. Because any algorithm won't test these edges first. The second part is the probability of each edge in exponential random graph is unknown. So here we set $\theta = \vec{0}$. So probability of every edge is $\frac{1}{2}$. Because we can set the cost of every edge arbitrarily, we can get the same result that the optimal first edge to test is the edge M from s to s_M if and only if there does not exist a cover of size smaller than k in the original set cover instance.

5 Algorithms

5.1 MDP-based Algorithm

I use Markov Decision Process(MDP) form an optimal algorithm. But since I have proved in former section, this problem is NP-hard. So this algorithm is time computational. There are two parts in this algorithm:

get the MDP formulation of this problem and use dynamic programming algorithm to get the optimal strategy π.

5.1.1 Mapping the problem into MDP

The key components of an MDP include decision epochs, state space, action sets, transition probabilities, rewards, decision policy and optimality criterion. This part is inspired by [5]. I make some changes in transition probabilities. So here I just introduce the state space, action sets, transition probabilities and rewards

- **State Space:** The state space of an MDP represents the possible states that a system can be in. It naturally corresponds to the set of temporary states S in this problem. We partition the state space S into $|E|$ disjoint subsets based on the number of edges having been tested in the states as $S = S_0 \cup S_1 \cup ... \cup |E|$. In decision epoch i, the system can only be in a state in S_i.

- **Action Sets:** For each state s 2 S, there is a set of actions that can be performed under it. We define the associated action set A_s of state s as the set of edges that have not been tested in s. Additionally, for terminating states, their action set also contains the terminating action $\perp$. As a result, the whole action set $A = \cup_{s \in S} A_s = E \cup \{\perp\}$.

- **Transition Probabilities and Rewards:** The transition probability is just the existence probability p(e),where e is the edge for next testing. The rewards is -c(e). But in this problem, we can't get the p(e) directly. Fortunately, we can calculate p(e) in current state by the probability of two after-state $s \cdot e$ and $s \backslash e$. First when we calculate P(Y=y) in exponential random graph model, we ignore the $K(\theta)^{-1}$. We will see this won't influence the final strategy. Suppose we have the $P(s \cdot e)$ and $P(s \backslash e)$ now for every un-testing edge e. Then we can get P(s) = $P(s \cdot e)$ + $P(s \backslash e)$. And $P(e|s) = \frac{K(\theta)^{-1} exp(\theta^T g(s \cdot e))}{K(\theta)^{-1} exp(\theta^T g(s \cdot e)) + K(\theta)^{-1} exp(\theta^T g(s \backslash e))} = \frac{exp(\theta^T g(s \cdot e))}{exp(\theta^T g(s \cdot e)) + exp(\theta^T g(s \backslash e))} = \frac{P(s \cdot e)}{P(s \cdot e) + P(s \backslash e)}$. So in the first epoch, since all edge in the state s all tested, we can calculate the P(s) directly. Then for the follow epoch, we already calculate the state probability which we need in this epoch in the last epoch, so we can calculate all state probability for state s in this epoch. The transition probability function is the same in [5](but we need do some extra calculation to get p(e)).

5.1.2 Dynamic Programming Algorithm

This part is the same as it in [5]. [5] proved that dynamic programming and this MDP formulation can lead to optimal adaptive testing strategies. The full algorithm is shown in **Algorithm 1**

Algorithm 1 The MDP-based algorithm

Input: Exponential Random Graph G(V, θ, g, c), source s, destination t
Output The optimal testing strategy π
Initialize: $u_\pi(s) = 0$, for all s $\in S_{|E|}$
for i = $|E|$ to 0 **do**
  for All s in S_i **do**
    if i = $|E|$ **then**
      p(s) = exp(θ^Tg(s))
    else
      p(s) = p(s·e) + p(s\e)
    if s is a terminating state **then**
      $u_\pi(s) := 0$, $\pi(s)$:=terminate
    else
      e^*:=arg $max_{e \in A_s}\{-c(e) + \frac{p(s \cdot e)}{p(s \cdot e)+p(s \backslash e)} u_\pi(s \cdot e) + \frac{p(s \backslash e)}{p(s \cdot e)+p(s \backslash e)} u_\pi(s \backslash e)\}$
      $u_\pi(s) := -c(e^*) + \frac{p(s \cdot e^*)}{p(s \cdot e^*)+p(s \backslash e^*)} u_\pi(s \cdot e^*) + \frac{p(s \backslash e^*)}{p(s \cdot e^*)+p(s \backslash e^*)} u_\pi(s \backslash e^*)$
      $\pi(s) := e^*$
return π

5.1.3 Computation Complexity

First there are $3^{\frac{|V|(|V|-1)}{2}}$ temporary state. Qualifying whether a state s is a terminating state can be realized by querying the s-t connectivity on two deterministic graphs $G_s^1(V)$ and $G_s^2(V)$. In $G_s^1(V)$ if s_e is 1, we add edge e into $G_s^1(V)$. In $G_s^2(V)$ if s_e is 1 or *, we add edge e into $G_s^2(V)$. This process needs O($|V|^2$) by Union-Find. Calculate p(s) is O($|V|^2$) because it need O($|V|^2$) time to find the features in defined. And selecting the optimal action for each state requires O($|V|^2$). So the total time complexity of this algorithm is $O(|V|^2 3^{\frac{|V|(|V|-1)}{2}})$.

5.2 Approximation Algorithm

Here I introduce two approximation algorithm, which both have polynomial time complexity. And the second approximation algorithm has a very good approximation guarantee.

5.2.1 Greedy Algorithm

The first approximation algorithm is a greedy algorithm which always choose the edge with smallest cost. Its time complexity is $O(n^2)$. And it's proved the approximation ratio is $O(\frac{|V|(|V|-1)}{2})$[5].

5.2.2 Intersection Sort Algorithm

This algorithm is proposed in [7]. It tests the edge with the minimum cost that lies on the intersection of a shortest s-t path and a minimum s-t cut in the uncertain graph under the current state. The basis of the third heuristic is the fact that we do not need to take the optimistic or pessimistic view since the intersection of any minimal path set and any minimal cut set is non-empty. The time complexity is $O(|V|^4)$ because it needs $O(|V|^2)$ to find the shortest s-t path by Dijkstra algorithm(it can be reduce to $O(|V| \log |V|)$) and $O(|V|^2)$ to find minimum s-t cut. And it tests at most $O(|V|^2)$ edges, so the final time complexity is $O(|V|^4)$. In simulations, this algorithm performs very close to optimal algorithm.

6 Simulations

The biggest problem in simulation is that we don't know the exact distribution of exponential random graph(unless we calculate all instance, it is impossible for a large network). So we can't get samples directly. Here I use Gibbs sampling algorithm to get the samples, then compare the optimal algorithm and the approximation algorithms.

6.1 Gibbs Sampling

Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. The key idea of MCMC is that is we choose a proper transition matrix, the states distribution will finally converge to the target distribution. So the observed data is the samplings which we want. And the idea of Gibbs sampling is to use the conditional distribution as the transition matrix.

In exponential random graph it is difficult to know the probability of an network instance, but it's easy to get the conditional distribution. The idea

is similar with the transition probabilities part in MDP former. If we seen each edge as a variable whose value is 0 or 1. Then our goal is to determine:

$$P(x_i|x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_{\frac{|V|(|V|-1)}{2}})$$

Here the x are all 0 or 1. For convenience we denote it as $P(x_i|...)$, where ... means all the other edges. Because the sum of probability of variable x_i to get 0 and 1 is 1. That is $P(0|...) + P(1|...) = 1$. We can calculate it in this way:

$$\begin{aligned} P(x_i|...) &= \frac{P(x_i|...)}{P(x_i|...) + P(1 - x_i|...)} \\ &= \frac{K(\theta)^{-1} exp(\theta^T g(y_{e_i=x_i}))}{K(\theta)^{-1} exp(\theta^T g(y_{e_i=x_i})) + K(\theta)^{-1} exp(\theta^T g(y_{e_i=1-x_i}))} \\ &= \frac{exp(\theta^T g(y_{e_i=x_i}))}{exp(\theta^T g(y_{e_i=x_i})) + exp(\theta^T g(y_{e_i=1-x_i}))} \end{aligned}$$

MCMC always needs a large iteration number to converge for exponential random graph. Because of the time limit, in the simulation I set the iteration number of Markov chain as 1500.

6.2 Simulation Parameters Setting and Results

In general a brain network is about 100 vertices. Because of the MDP-based algorithm has a high time complexity, I run simulations on a small network(only five vertices) with the MDP-based algorithm and the approximation algorithms and a large network(90 vertices) with only the approximation algorithms. The large network use 720 samples and randomly choose 6 s-t pairs. The small network use all possible samples and randomly choose 6 s-t pairs.

For the exponential random graph model I use the 4 most important metrics: edges, GWD, GWESP, GWNSP. The corresponding θ is [-4.48,1.51,-0.15,1.12]. These parameters come form [3]. Author fits the model by MCMC MLE from data of 10 volunteer's brains. The cost is Gaussian distribution with mean 50 and standard deviation 10. Figure 5 and Figure 6 show the results.

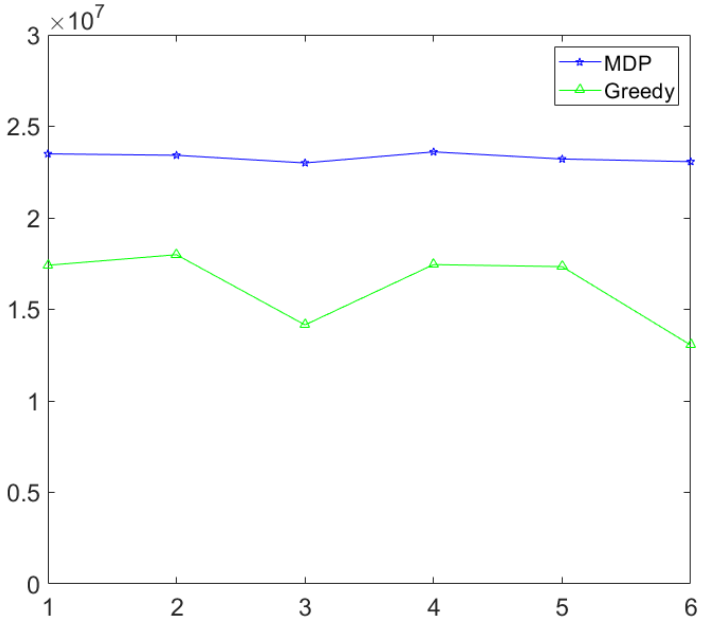

Figure 5: Simulation on a large network

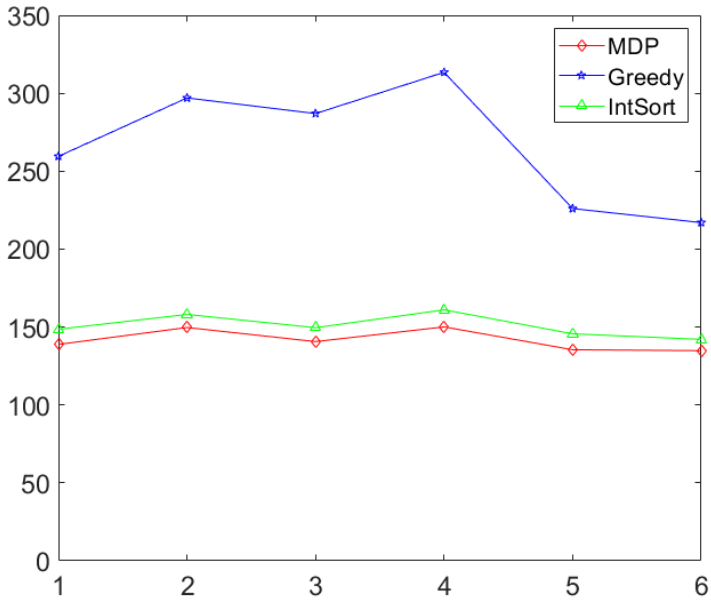

Figure 6: Simulation on a small network

From figure 5 we can see the Intersection sort algorithm performs better than greedy algorithm. From figure 6 we can see that Intersection algorithm is very close to the optimal algorithm. Because it is difficult to get the real brain data, I just run the algorithm on the network model instead of a real brain network.

7 Conclusion

In this paper, we modeled the brain network into a exponential random graph which determinate the probability of every instance by some metric in the network. Assuming that during each determining process we are associated with an underlying graph, the existence of each edge can be unraveled through edge testing at a cost of c(e). Then we proposed the problem to find the optimal strategy incurring the minimum expected cost. And we proved this problem is NP-hard both holistically and sequentially. Then we give a MDP-based optimal algorithm and two approximation algorithms. We analysis the detail and the time complexity of these algorithms.Finally we make two simulation to see the simulation results and to compare these three algorithms.

8 References

1. E.W.Lang,1 A.M.Tome,I.R.Keck,J.M.Gorriz-Saez,and C.G.Puntonet, Brain Connectivity Analysis: A Short Survey, Computational Intelligence and Neuroscience , 2012, 2012(4):412512

2. D.J.Watts and S.H.Strogatz, Collective dynamics of 'smallworld9 networks, Nature, vol. 393, no. 6684, pp. 440442, 1998.

3. Sean L.Simpson,Satoru Hayasaka , Paul J.Laurienti, Exponential Random Graph Modeling for Complex Brain Networks. arxiv:1007.3230

4. Hunter DR, Goodreau SM, Handcock MS (2008) Goodness of fit of social network models. Journal of the American Statistical Association 103: 248-258.

5. Xinzhe Fu, Zhiying Xu, Qianyang Peng, Luoyi Fu, Xinbing Wang, Complexity vs. Optimality: Unraveling Source-Destination Connection in Uncertain Graphs . IEEE INFOCOM 2017

6. M. O. Ball, Computational Complexity of Network Reliability Analysis: An Overview, in IEEE Trans. on Reliability, Vol. 35, No. 3, pp. 230-239,1986

7. L.A. Cox Jr., Y. Qiu, W. Kuehner, Heuristic least-cost computation of discrete classi7cation functions with uncertain argument values, Ann. Oper. Res. 21 (1989) 121.