IMPERIAL COLLEGE LONDON

Link prediction in insect networks

Zhenbo Zhu
Supervised by Dr. Ray Prasun

September 2021

List of Contents

1 Introduction

Link prediction is a very important topic in network analysis, as it provides information about how the network would evolve. It also has a very wide application in different fields. There are many studies about link prediction [8], and they perform well in specific networks. Many of them have a nice performance in human networks and even animal networks, but we have no idea how these methods perform in insect networks. Besides, the mentioned networks are all static networks without information about the time of the formation of links. In this report, we aim to apply different static link prediction methods on temporal insect networks and find how well each method works on insect networks in each stage. Finally, we design a new method inspired by [5] which gives different score of links as time varies to try to explain how links are added to the network at each time and use it to predict potential links in the future.

In the second section, we will introduce the two networks we are going to predict as well as some basic properties (degree distribution, average degree and average clustering). Then in the third section, we will describe the methods used in link prediction including similarity based methods and probability inference models. Afterwards, we will describe how to evaluate the link prediction methods (AUC, precision and some metrics associated with time). In the fourth section, we show our results and based on some results, we will fit the collected data roughly and give a very simple score to each node pair. At the end of the section, we also perform network reconstruction to see if some important properties would be preserved with some of link prediction methods. Finally, we will conclude the report and end it with some discussion of direction of further studies.

2 Data

2.1 Bee network

The data is from [4] where the experiment collect interaction of four groups of bees with three types of different interactions. Here, we choose the group with most bees with the contact interaction type although according to the paper, the contact interaction plays a far more important role in information transmission than the other two types of interaction. This is because we mainly focus on the evolution of network and the contact interaction can provide much more data (about 6 times larger).

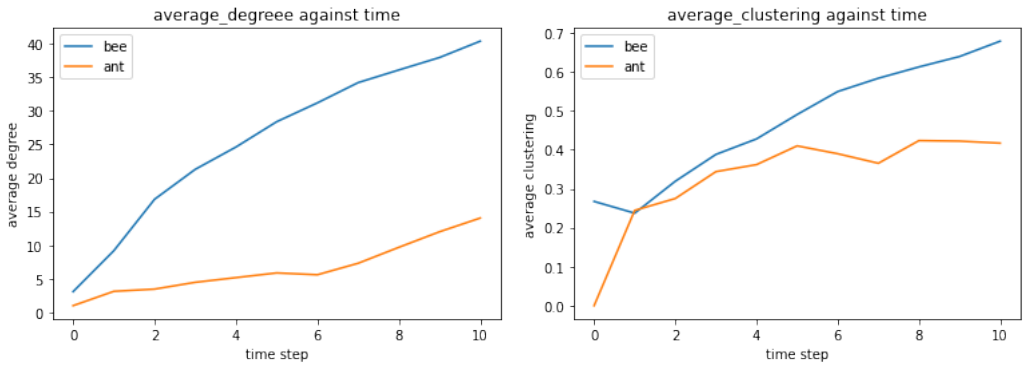
The network has 69 nodes(bees) with 4197 edges(times of interaction). If we simplify it as a simple graph without multiple edges and self loops, it will have 1393 edges. The average clustering is 0.68 and the average degree is 40.38.

The time when interactions happen is recorded and therefore, we can have a plot that shows how the key properties change against time.

2.2 Ant colony

The data is from [2] and is available in R timeordered package[1]. There are ants and interactions between them during about 1800s time span. If we simplify it to a simple graph, edges would be left. The average degree is 14.58 and the average clustering is 0.43.

To compare these two colonies, we put these the changing average degree and average clustering in a same graph and we define 800 as a timestep in bee network and 50 in ant network:



Here is the degree distribution of these two networks:

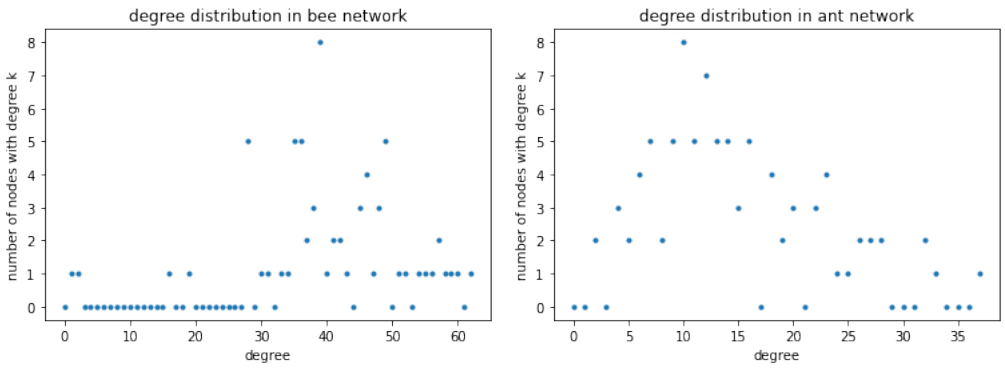


Figure 1: degree distribution of bees (left) and ants (right)

3 Methodology

3.1 Similarity Based Algorithm

To predict links in a network, we may intuitively connect two nodes according to their properties, these properties can simply be ages, interests or mutually information like degree product of two nodes. Furthermore, it can be local information such as, distance between the two nodes and number of common neighbours. It can even be some global information like Katz centralities.

In this report, we use local similarities including common neighbour similarity[11], two walktrap similarities[9] and degree product similarity[5]. Then we use the Katz similarity[7] as a global similarity.

3.1.1 Degree product similarity

It is simply defined by the product of degree of two nodes. It is found that many of networks are scale-free network with a clear preferential attachment that edges tend to be add to nodes with higher degrees. However, we can see from Fig.1, neither these two networks are scale-free network. Besides, we plot the cumulative probability of edge being put between degree product of two nodes. And it is not a straight line as described in Barabasi-Jeong's Paper, instead, it increases much slower after degree product reaches some certain value. Further discussion are in the 4.1.4 section.

3.1.2 Common neighbour similarity

It is the number of common neighbours between two nodes: $S_{CN}(x, y) = |\Gamma(x)| \cap |\Gamma(y)|$ where $\Gamma(x)$ is the set of neighbours of x. Newman conjectures that two nodes with more common neighbours are more likely to be connected, we plot the probability that edge is added to different number of common neighbours and find an increase and then decrease curve. Although Newman also gets such a curve(Fig.2) and explains this by the fact that nodes with so many common neighbours are rarely not connected[11]. However, they are not the same case. We can see that in Newman's plot, the decrease rate is obviously slower than increase rate, high common neighbour pair still has a relatively high probability to be connected, but in this plot, the decrease rate is also very fast.

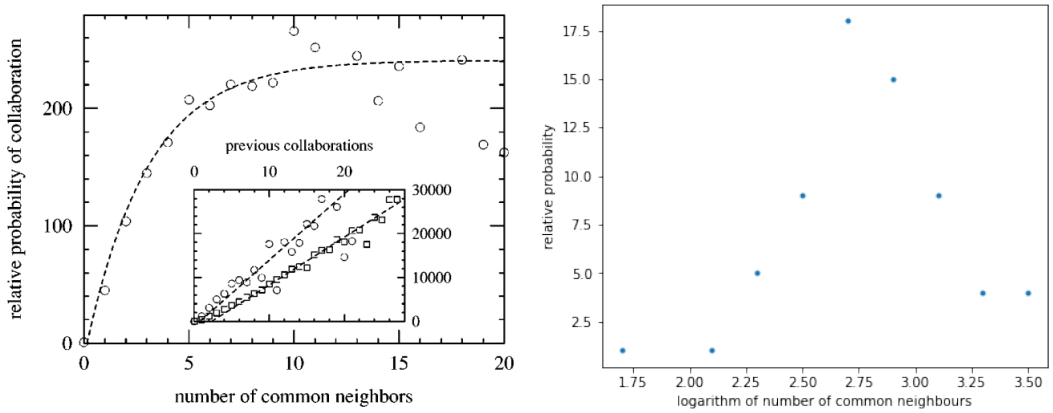


Figure 2: The left is Newman's plot[11] and the right one is our result.

There are also many local similarities like Jaccard Similarity, Salton Similarity and so on. However, since they are similar to common neighbour similarity with a difference of normalization.

3.2 Global Similarity

3.2.1 Local Random Walk Similarity

This similarity is the probability of reaching a node from another node through random walk in certain steps.[9] Say, starting at a node x, every time step, an ant will move to its neighbour uniformly randomly, i.e a_{xy}/k_x where a_xy is the xy entry of adjacency matrix and k_x is the degree of node x. Hence, the probability that the ant reaches other nodes at time step t is:

$$\vec{\pi_x}(t) = P^T \vec{\pi_x}(t-1)$$

Where $p_xy = a_{xy}/k_x$ is the probability of a single move mentioned above. Similarly, we will have an ant starting at y and thus, we define the walktrap similarity of time step t between node x and y:

$$S_t^{LRW}(x,y) = \frac{k_x \pi_{xy}(t)}{2|E|} + \frac{k_y \pi_{yx}(t)}{2|E|}$$

$|E|$ is the number of edges and we can easily see that $S_t^{LRW}(x,y) = S_{WT}(y,x)$. Here, an degree-related initial configuration function $k_x/2|E|$ is used and in the stationary state, it is equivalent to the degree product similarity.

However, Liu and lü point out that, processing this random walk, an ant would go to somewhere far away from the nodes, but most real networks have high clustering and random walkers are more likely to circulate locally. To fix this, Liu and Lü continuously release the walkers at the starting point, resulting in a higher similarity between close node pairs called superposed random walk(SRW):

$$S_t^{SRW}(x,y) = \sum_{s=1}^{t} S_s^{LRW}(x,y)$$

.

3.2.2 Katz Similarity

Katz Similarity focus on number of different paths between two nodes with a constant β representing the probability of effectiveness of a single link. Therefore, longer paths will contribute exponentially less to the metric[7]. Briefly, the Katz similarity between two nodes x and y can be expressed as

$$S_{Katz}(x,y) = \sum_{1}^{\infty} \beta^n a_{xy}^n,$$

where a_{xy}^n is the $x-y$ entry of nth power of adjacency matrix, i.e the number of paths of length n between x and y. We can see that, for very small β, the similairty would be similar to the common neighbour similarity as the longer path would contribute to the similarity very less. In matrix form, it can be written as

$$S_{Katz} = (I - \beta A)^{-1} - I.$$

Note that, to ensure the convergence of the geometric series, the β must be smaller than the reciprocal of the largest eigenvalue of A.

3.3 Maximum Likelihood Methods

3.3.1 Hierarchical Structure Model

Hierarchical structure is detected in many networks, and we can display this structure through dendrograms with n-1 internal nodes. Each internal node is associated with a probability that the left and right subtrees are connected. Clearly, the probability that two nodes are connected is the probability associated to the lowest common ancestor of these two nodes. Therefore, with the best fit hierarchical random graph (hrg), we can easily make prediction of potential links. Given a network G, the likelihood of a hierarchical random graph D is

$$P(D, p_r|G) = \frac{P(G|D, p_r) \times P(D, p_r)}{P(G)} = C\ L(D, \{p_r\}) = C \prod_r\ p_r^{E_r}(1-p_r)^{L_r R_r - E_r},$$

where r is an internal node of hrg and E_r the the number of edges in network G whose endpoints have r as their lowest common ancestor. C is a constant if we assume that each hrg with corresponding set of p_r is equally likely in prior. Besides, L_r and R_r are the number of nodes in the left and right subtrees of r respectively. If we fix D and take derivative of $L(D, \{p_r\})$, we can easily get $\bar{p_r} = \frac{L_r R_r}{E_r}$ that maximises the likelihood. Then we plug $\bar{p_r}$ and compute the log-likelihood:

$$\log L(D) = -\sum_r L_r R_r h(\bar{p_r}),$$

with $h(p) = -p\log(p) - (1-p)\log(1-p)$ the Gibbs-Shannon entropy function.

Then starting at a hierarchical random graph, we can randomly choose an internal node of D, then randomly reorder the three subtrees associated with it as shown in Fig.3. If the maximum log-likelihood of new hrg does not decrease, the transition will be accepted, otherwise if the log-likelihood decreases, the transition is accepted with probability $exp(\Delta\ log\ L)$. Repeating this process until equilibrium is achieved will give the fit hrg.

Figure 3: The three possible cases that an internal node r is placed. Figure is from [3]

To make prediction, we sample large number of dendograms from the fit hrg following the process described above, compute the probability of missing link and take average over the sampled dendrograms.

3.3.2 Stochastic Block Model

Stochastic block model is used to detect communities in a network[6][12][14]. It can be displayed by giving each node a number corresponding to which community it belongs to. This model assumes that nodes in the same community are equally likely to be connected with other nodes. For example, a_1 and a_2 are in the same community A, then not only is the probability of a_1 being connected with other nodes in community A and a_1 the same as that of a_2, but the probability of a_1 being connected with nodes in other communities as well. Besides, unlike other prediction methods, this method can take account into multiple edges. So we let ω_{ij} be the expected value of ijth entry of the adjacent matrix. It is clearly that the value of ω would be the same if the pair of nodes are in the same

community as the other pair respectively, so it would be convenient to write ω_{rs} representing the expected value of nodes in community r and s respectively. We will not consider self-loops, as they will not appear in our two networks. Another assumption is that the number of edges between two nodes follows Poisson Distribution. Thus the probability that the network is generated by ω_{ij} and b_i (the community that node i belongs to) is:

$$P(G|\omega, \mathbf{b}) = \prod_{i<j} \frac{(\omega_{b_i b_j})^{A_{ij}}}{A_{ij}!} e^{-\omega_{b_i b_j}}.$$

Since $A_{ij} = A_{ji}$ and $\omega_{b_i b_j} = \omega_{b_j b_i}$, we may rewrite it as:

$$\prod_{i<j} \frac{1}{A_{ij}!} \times \prod_{rs} (\omega_{rs})^{m_{rs}/2} e^{-\frac{1}{2} n_r n_s \omega_{rs}},$$

where $m_{rs} = \sum_{ij} A_{ij}\delta_{b_i r}\delta_{b_j s}$ represents the number of edges between community r and community s or twice the number of edges if $r = s$ and n_s is the number of nodes in community s. Taking logarithm and neglecting constant independent of ω and b yields:

$$log\ P(\omega, \mathbf{b}) = \sum_{rs} m_{rs} log(\omega_{rs}) - n_r n_s \omega_{rs}$$

.

Similar to the hierarchical structure model, we differentiate the log likelihood and obtain the $\bar{\omega_{rs}}$ that maximises the log-likelihood: $\bar{\omega_r} s = \frac{m_{rs}}{n_r n_s}$. Plug this into the equation above and we get

$$L(G|\mathbf{b}) = \sum_{rs} m_{rs} log(\frac{m_{rs}}{n_r n_s})$$

.

However, placing edges uniformly randomly in each community would cause the nodes in the same community tend to have similar degrees. To fix this problem, degree-correct models may be used. This model has an extra set of parameters $\{\theta_i\}$ that controls the expected degree of node i. Now, the expected ijth entry of adjacent matrix is $\theta_i \theta_j \omega_{b_i b_j}$. Thus, the likelihood becomes:

$$P(G|\omega, \mathbf{b}, \theta) = \prod_{i<j} \frac{(\theta_i \theta_j \omega_{b_i b_j})^{A_{ij}}}{A_{ij}!} e^{-\theta_i \theta_j \omega_{b_i b_j}}.$$

We may assume that $\sum_i \theta_i \delta_{b_i, r} = 1$ for all groups r, since a multiplicative constant can be absorbed by ω. With this constraint, we may rewrite the equation as:

$$P(G|\omega, \mathbf{b}, \theta) = \prod_{i<j} \frac{1}{A_{ij}!} \times \prod_i {\theta_i}^{k_i} \prod_{rs} (\omega_{rs})^{m_{rs}/2} e^{-\omega_{rs}/2},$$

where k_i is the degree of node i

As before, we take logarithm of the likelihood and ignore independent constant:

$$log\ P(G|\omega, \mathbf{b}, \theta) = 2 \sum_i k_i log\ \theta_i + \sum_{rs} (m_{rs} log \omega_{rs} - \omega_{rs}).$$

Use similar method as before, we may compute the $\bar{\theta}_i$ and $\bar{\omega_{rs}}$ maximise the log-likelihood:

$$\bar{\theta}_i = k_i/K_{b_i},\ \bar{\omega_{rs}} = m_{rs} \text{ ,with } K_r = \sum_i k_i \delta_{r,b_i}.$$

This model preserves the expected degree for each node:

$$\sum_j \langle A_{ij} \rangle = \bar{\theta}_i \bar{\theta}_j \bar{\omega_{b_i b_j}} = \frac{k_i}{K_{b_i}} \sum_j \frac{k_j}{K_{b_j}} m_{b_i b_j} = \frac{k_i}{K_{b_i}} \sum_j \sum_r \frac{k_j}{K_r} m_{gi,r} \delta_{gj,r} = \frac{k_i}{K_{b_i}} \sum_r m_{b_i r} = k_i.$$

Back to the log-likelihood, substituting $\bar{\theta}_i$ and $\bar{\omega_{rs}}$ and doing some simplification yields:

$$\log\ P(G|\omega, \mathbf{b}, \theta) = \sum_{rs}\ m_{rs} log(\frac{m_{rs}}{K_r K_s}).$$

With this objective function, we can perform similar process as we do in hierarchical structure model: starting from randomly K groups, we move a node into another community and compute the change of log-likelihood. According to [6], it is easy to compute it in time $O(K + \langle k \rangle$.

However, the drawback is very obvious that the best partition can be computed only with respect to a certain number of partitions. This is very annoying because it is very hard to decide how many partitions should be divided into. Indeed, the accuracy of partition would increase as the number of partition increases. Therefore, the non parametric Beyasian inference is introduced by [14]. The basic idea is to maximise the full joint probability of the network and all parameters to maximise the posterior likelihood. This is equivalent to minimise the description length Σ of the data[1]:

$$\Sigma = -\log_2 P(A, \mathbf{k}, \mathbf{e}, \mathbf{b}) = S + L,$$

where $S = -\log_2 P(A|\mathbf{k}, \mathbf{e}, \mathbf{b})$ is the number of bits necessary to precisely describe the network, and $L = -\log_2 P(\mathbf{k}, \mathbf{e}, \mathbf{b})$ is the number of bits necessary to describe the model parameters. The computation is complicated and it can be found in [14].

The model would avoid overfitting because as the number of blocks increases, the L will increase but the S will decrease.

Then, similar process can be performed to minimise the description length and find the as good partition as possible. [2]

The selection of degree-corrected model and non degree-model depends on the networks. A method helps to make decision is to find the ratio of the joint posterior probability $P(\mathbf{b}, H_{DC}|A)$ with $H_{(N)DC}$ being the hypothesis of (non) degree-corrected model:

$$\begin{aligned}\Lambda &= \frac{P(\mathbf{b}, H_{NDC}|A)}{P(\mathbf{b}, H_{DC})|A} \\ &= \frac{P(A, \mathbf{b}, H_{NDC})}{P(A, \mathbf{b}, H_{DC})} \times \frac{P(H_{NDC})}{P(H_{DC})} \\ &= 2^{-\Delta\Sigma}\end{aligned}$$

[1] Here, the number of edges between communities and the degree sequences are used as a parameter instead of θ and ω we used above. In [14], these two likelihoods are proved to be equivalent in most networks (except for small or sparse networks).

[2] The algorithm implemented in graph-tool is more complicated for efficient purpose and this is beyond the scope of discussion. More details of the algorithm can be found in [12].

If $\Lambda < 1$, the degree-corrected model is preferred, and the non-dgree-corrected model is better if $\Lambda > 1$.

3.4 Evaluation Method

It is important to evaluate these link prediction methods, here we use three methods(area under the receiver operating characteristic curve(AUC), precision and precision regarding network evolution).

3.4.1 AUC and precision

Let the set of observed edges be E, and then E_c the complement of E with the set of all possible links in the network as the universe set stands for the potential links of the network. We randomly divide E into two partitions probe set E_p and training set E_t. Now we want to use the information of E_t to try to make prediction.

The AUC calculates the probability that a false negative (neither in E_t nor in E) is ranking lower than true negative (not in E_t but in E). To compute this, we randomly choose q links in E_P and E^c, then count the number of links from potential links that have lower score than and same score as links from probe links, denoted as m and m' respectively. Then the AUC is achieved by:

$$AUC = \frac{m + 0.5m'}{q}.$$

The precision counts the proportion of true negative among top-L links. In our analysis the L is just the number of links in probe links. To compute precision, we get top-L links with highest score and let p be the number of links in probe links among the L picked links. Then the precision is the ratio of p and L:

$$precision = \frac{p}{L}.$$

3.4.2 Precision and AUC Regarding Network Evolution

We write this as p_t and it is helpful to analyze our networks since our network has time of interaction. Thus, we denote $L_{t,\Delta t}$ the number of links adding between time $t,\ t + \Delta t$, and let $p_{t,\Delta t}$ be the number of new added links having top $q_{t,\Delta t}$ scores. It is very similar to precision if we regard the new added links as the probe links with $L = L_{t,\Delta t}$. It is the proportion of links being predicted correctly during time t to $t + \Delta t$:

$$p_t = \frac{L_{t,\Delta t}}{q_{t,\Delta t}}$$

Similarly, we pick q links from new added links during the time period $t, t + \Delta t$, compare their score to the q randomly chosen missing links and record the number that the former has higher score as m and equal score as m'. Thus the AUC regarding network evolution is:

$$AUC_t = \frac{m + 0.5m'}{q}.$$

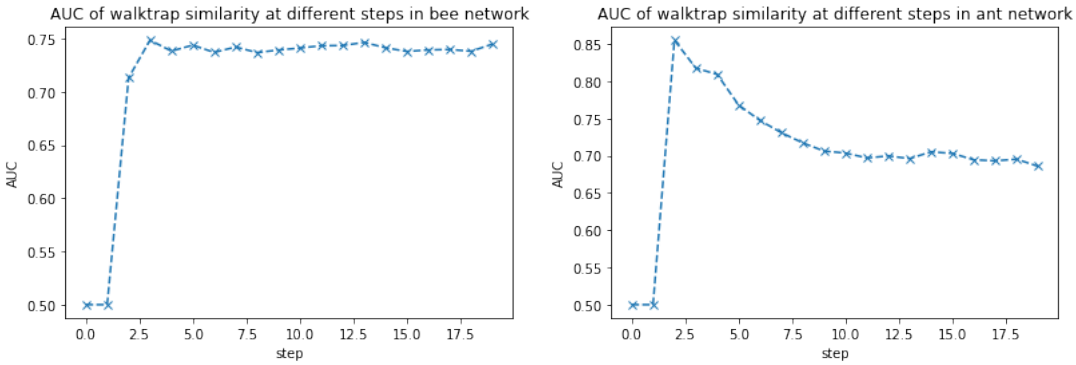
4 Data Analysis and Discussion

This section mainly displays the result of analysis based on section 3.1,section 3.2 and section 3.3 It consists of three subsection. We firstly evaluate these link prediction methods using metrics we discussed above. Then we find out how links are added to networks related to degree product of two nodes and use this to adjust degree product similarity to give a better performing link prediction method. Finally, we try to reconstruct the network with the help of the hrg models

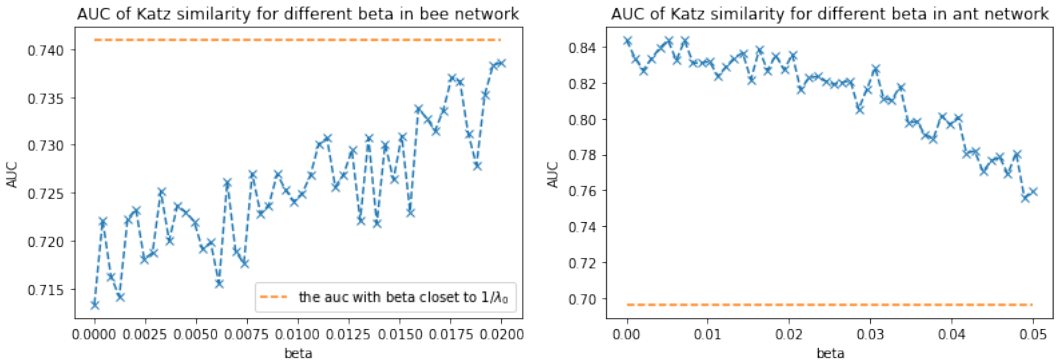
4.1 Evaluation of Link prediction Methods

4.1.1 Some Preparation

Before we compute the AUC, we may determine how many time steps should be chosen in LRW random. For this purpose, we plot the AUC of LRW with 10% edges being removed at different time steps:



Then, the β of Katz similarity should also be determined. Therefore, we plot the AUC of Katz similarity with different β. The dashed yellow line is the AUC of $beta$ closed to the reciprocal of the leading eigenvalue of adjacency matrix:



We can see that in bee networks, the β closer to $1/\lambda_0$ gives higher AUC but in ant networks, it becomes largest when β is about 0.1.
Besides, whether the degree-corrected model should be chosen is also important. We compute the entropy and the difference ($\log \Lambda$ described in 3.3.2)[3]:

First is the bee network, and the second is the ant network. We can see that degree-corrected model is preferred in bee network and the case is opposite in ant network. 27 seems not very

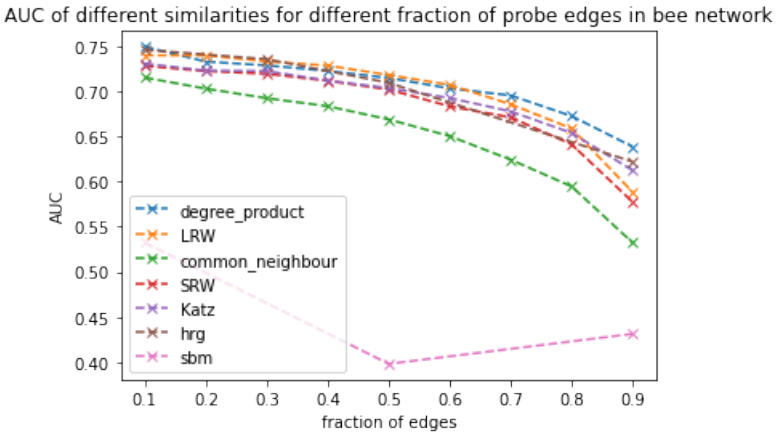
[3]It may be possible that the same network admits many alternative fits with very similar posterior probabilities. Therefore, it would be better to compute the ratio of posterior probability of the entire model, but since here a clear result is given, we would not do this. Detail of this method can be found in [14]

```
Non-degree-corrected DL:          4152.805762261131
Degree-corrected DL:      4125.007412080152
ln Λ:  27.79835018097947
Non-degree-corrected DL:          3178.1227435163596
Degree-corrected DL:      3234.0547127889004
ln Λ:  -55.931969272540755
```

large but it actually means that the posterior probability of degree corrected model is about e^27, i.e. 5×10^11 times larger than that of non degree corrected model.

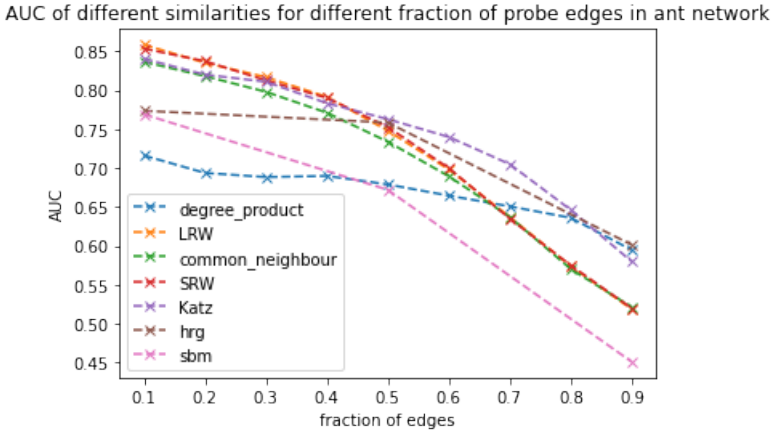
4.1.2 AUC

The following graph is the AUC of different methods in bee networks. The horizontal axis is the fraction of edges to be removed (i.e size of probe set) and the vertical axis is the value of AUC. The β of Katz similarity is set to be closer to $1/\lambda_0$ in bee network but is set to be 0.01 in ant network. The step of LRW similarity is chosen to maximise the AUC and the SBM model is chosen as described in section 3.4.



From the graph, we can see that LRW similarity HRG method and degree product similarity perform better than others in bee network but overall the SRW and Katz similarity also have a nice performance. Common neighbour similarity also does not have a poor performance. However, the SBM meodel seems not very suitable to this network.

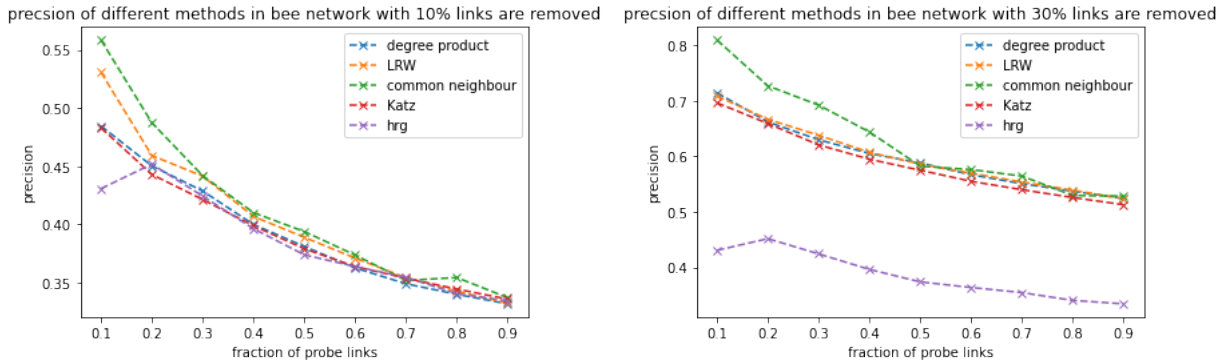
Then we look at the AUC in ant network:



In ant network, SRW and LRW have very high score (more than 0.85) when 10% edges are removed. The Katz similarity and common neighbour similarity also have a very close AUC to SRW and LRW. The hrg and sbm method have a not very bad score and the degree product similarity has a lowest and not bad AUC value.

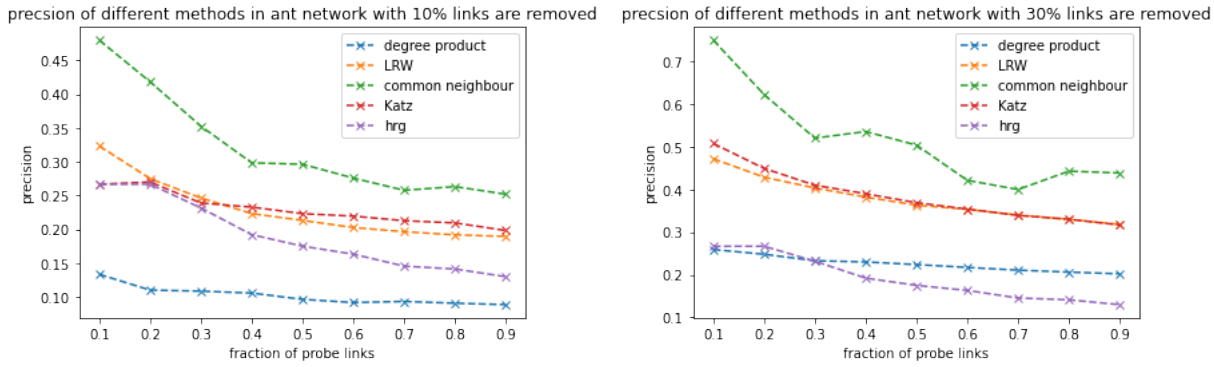
4.1.3 Precision

Now we need to look at the precision of these methods to have a further judgement on these methods. First is the bee network. The horizontal axis is the fraction of the probe set to be chosen as L, and portion of top-L links that is in probe set is the precision. In the left graph, 10% of edges are removed as probe set and 30% is removed in the right graph:



The patterns of most methods are very similar in the two graphs: common neighbour similarity performs best and then followed by LRW, and hrg performs worst. But from left graph to the right one, we can see that the precision value obviously increase. This means that when more links are removed from the graph, the portion of links having high ranks increases faster. When 30% of links are removed, the $40(0.1 \times 417)$ highest links are predicted very accurately (more than 80%).

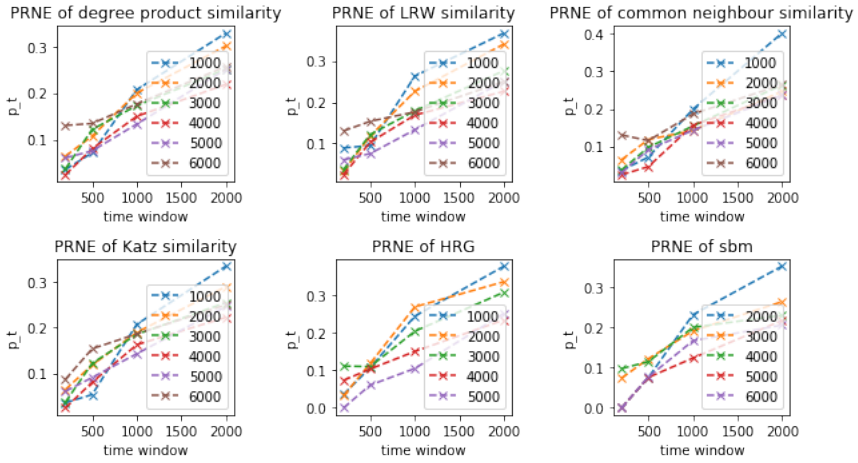
And this is the precision in ant network:



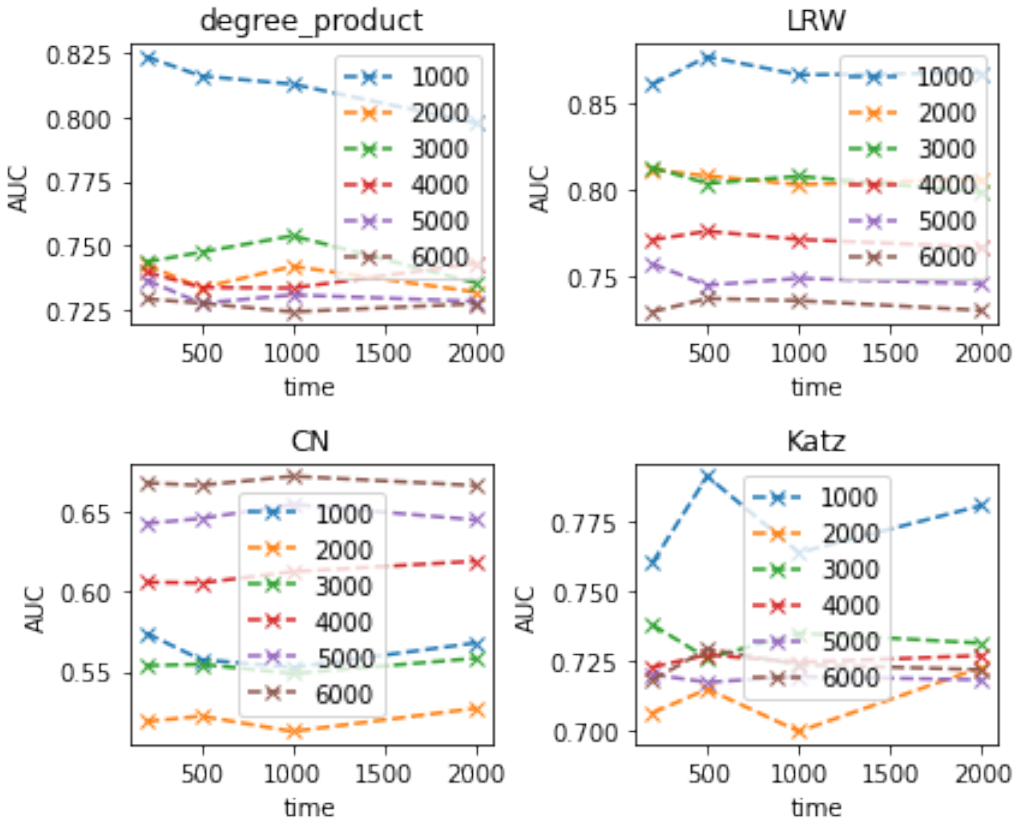
These methods perform very similarly. Common neighbour has the highest precision but degree product similarity is the worst when 10% links are removed.

4.1.4 Precision and AUC at Different Time

The prediction methods all have a not bad performance on the static network, but for a temporal network, it is also interesting to see how well these methods work on predicting the coming links. Suppose we have L links added during the period t to Δt, we calculate the portion of top L links that are added in this time period. p_t at different time with different time window are plotted:



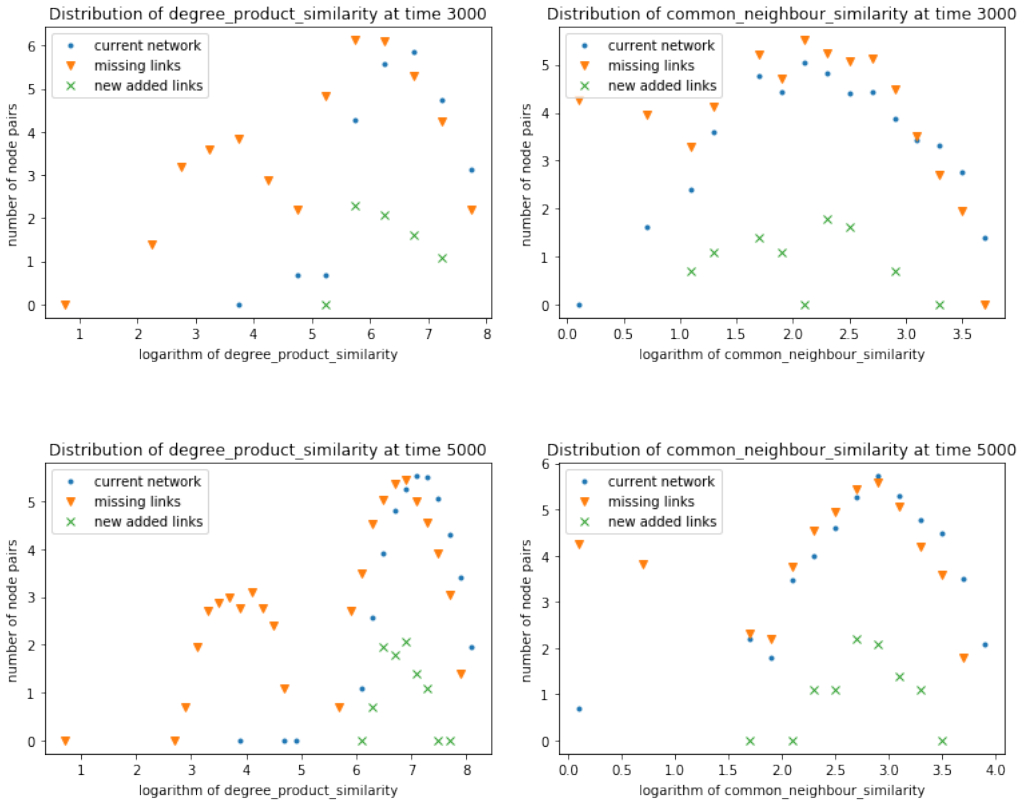
Besides, AUC at different time is also plotted:



We can see that the portion is not high even for a long time span. but is it the AUC is high as network evolves. Since the AUC samples from static network, it relies a lot on network structure. This gives information about how network evolves but it works better in stable networks. But in the early stage of network, it may make problematic conclusions. It is possible that in early stage of a network, there are many links with low degree product and few links with high degree product. Even if higher degree product pairs are more likely to be connected, we still get a poor AUC. In this case, AUC does not correctly evaluate the method although as time goes, more links with higher common neighbours would appear in the network due to a higher increasing rate. Therefore, AUC varies with time for these similarity based methods. Besides, it may also be possible that the manner of links being connected may also change, so it is interesting to see how edges are added at different

time.

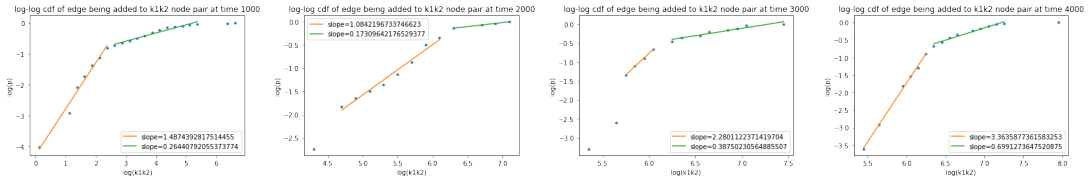
Below is the log=log degree product distribution and common neighbour distribution in current networks, missing edges and new added edges in the following $200s$ at $2000s$ and $5000s$. We can see that

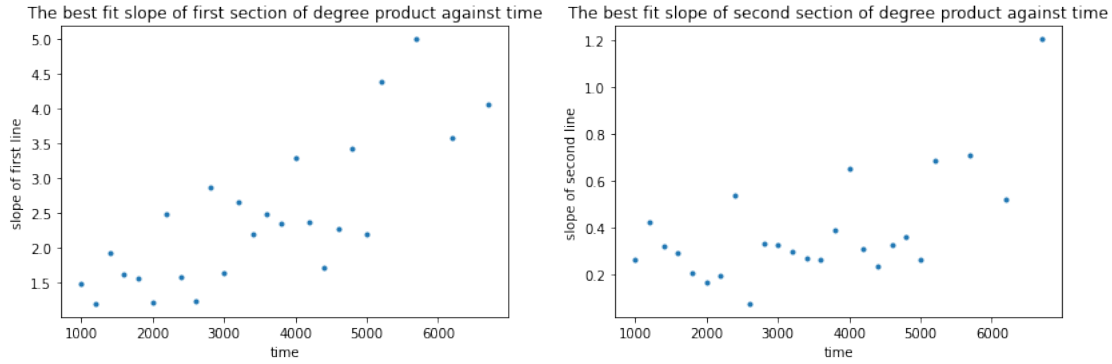


The left two are degree product similarity and the right two are common neighbour similarity. We may see that the AUC of degree product similarity would be high because there are larger portion of missing links having low degree product but the most existing links have a high degree product. For similar reason, the AUC regarding network evolution would also be high. This reveals that high degree product node pairs are more likely to be linked, not not the highest. Thus our precision is not very high as top links are not very likely to be added. This will cause if we add enough many links to contain the peak as the top L links, the precision would be very high as we observed, but it would be hard to make an accurate prediction of the following added few links. Therefore, it may be helpful to adjust the similarity and try to make the peak have highest similarity.

A possible way to achieve this is to plot cumulative probability of an edge is added to node pairs with degree product less than or equal to a certain value from time t to $t + \Delta t$, and then taking derivative of the c.d.f yields the probability of certain degree product node pair to receive links. In Barabasi and Jeong's example, the curve should be a straight line, but in our bee network, the slope plunges when degree product reaches some value. To find more about this, we also plot the slope of first and second part of the curve against time and also the evolution of the changing point:

From time $1000s$ to $5000s$, the time window is $200s$, and then from $5200s$ to 6700 the time window is 500. We can see that the curve with respect to the slope frustrated around 1.5 before 3000s, after that the slope is around 2.5 from $3200s$ to $5000s$. After that, it increases to about 3.5 to 4.(I tend to





mark the two high slopes as error) In contrast, the second slope is more stable around 0.3 before $5000s$ and increase very fast after that.

Therefore, We may fit a function for each of these three parameters, say $a(t)$, $b(t)$ and $c(t)$ respectively. Then, The probability that a link is added to two nodes with degree product less than or equal to $k_i k_j$ is

$$\int_0^{k_i k_j} \pi(k_i k_j, t) dk_i k_j = \begin{cases} C(t)(k_i k_j)^{a(t)}, \text{ if } \log(k_i k_j) \leq c(t) \\ \frac{c^{a(t)} e^{C(t)}}{c(t)^{b(t)}} (k_i k_j)^{b(t)}, \text{ if } \log(k_i k_j) > c(t). \end{cases}$$

Therefore, the new degree product similarity becomes:

$$\pi(k_i k_j, t) = \begin{cases} C(t)(k_i k_j)^{a(t)-1}, \text{ if } \log(k_i k_j) \leq c(t) \\ \frac{C(t) c^{a(t)}}{c(t)^{b(t)}} (k_i k_j)^{b(t)-1}, \text{ if } \log(k_i k_j) > c(t). \end{cases}$$

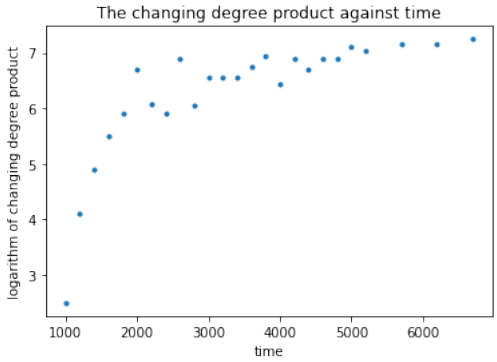
In our work, we simply fit these three curve by[4]:

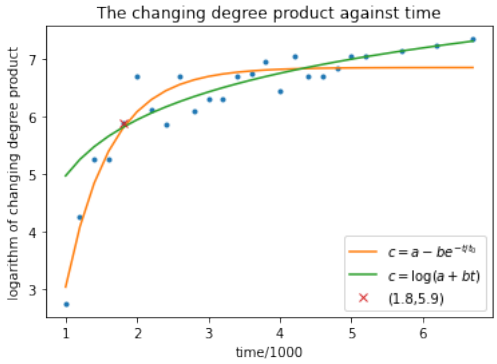
$$\begin{cases} a(t) = \begin{cases} 1.5, & \text{if } 1000 \leq t < 3000 \\ 2.5, & \text{if } 3000 \leq t < 5000 \\ 4.0, & \text{if } 5000 \leq t \leq 7000 \end{cases} \\ b(t) = 0.3 \\ c(t) = \begin{cases} A_1 - B_1 e^{-t/t_0}, & \text{if } 1000 \leq t \leq 1800 \\ \log(A_2 t + B_2), & \text{if } t > 1800 \end{cases} \end{cases}$$

The fit of $c(t)$ looks like this:

We expect this similarity would have a high accuracy at different time. Indeed, we adjust this according to how new edges are added at time t to $t + \Delta t$. It may be better if we separate it into three parts because in some curves have a steeper slope at the beginning but the first part in this case usually consists of two to three points and at most time the first two slopes of the three are the

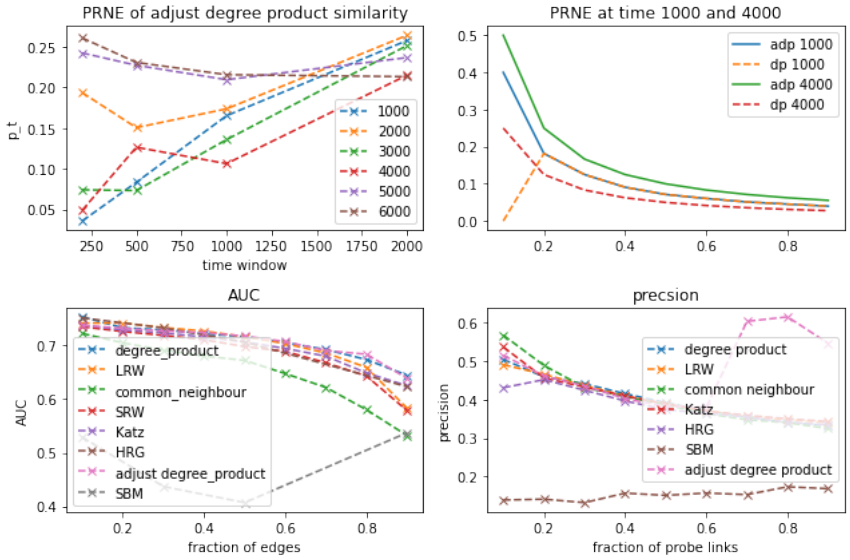
[4] It is obvious that the fit is very rough work due to time limit. However, for simple prediction purpose, the value of a and b is not very important.





same, so we only use two lines to fit the curve for convenience.

We can see that, the PNRE doubles or even more at different time. Even at some time when the adjust degree product similarity does not outperform like $1000s$ and $4000s$, we can see that it performs much better than degree product similarity for the very top few links at these time (the second figure). It also has a very nice performance at the final network with a high AUC and precision:

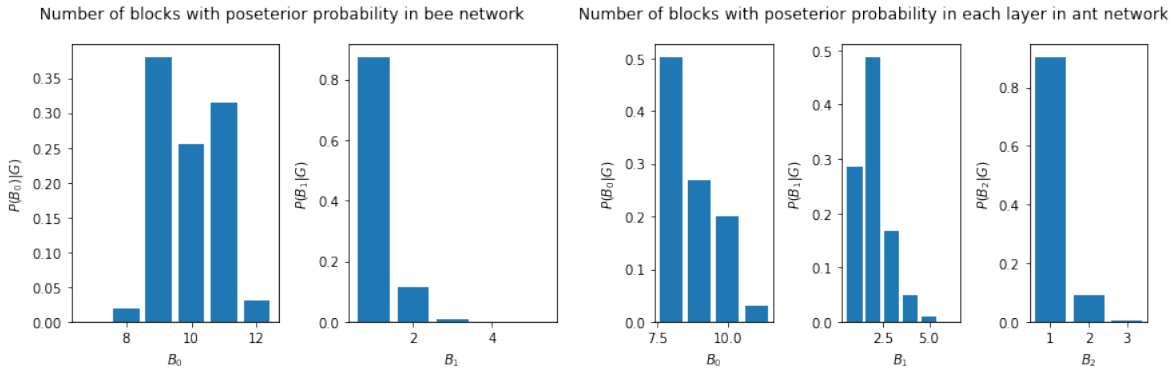


4.2 Some Other Results

Except for link prediction, the HRG and SBM methods also include much information of the network. This section would talk about the reconstruction of the network based on HRG method and the hierarchical community structure inferred by SBM model.

4.2.1 Community Structure

Based on the method described in 3.3, we infer the community structure of bee and ant network:



The left is the posterior probability of number of communities in bee networks and the right is that of ant. We can see that 11 to 13 communities are inferred and they converge to one community rapidly. This shows that there may be no hierarchical structure in bee networks and the community is also weird. According to [4], the experiment is designed to find out how interactions between bees change their forage. So we may guess that only worker bees are picked instead of a whole colony. In contrast, ant colony has a clear hierarchical structure, and small community emerge to two bigger communities. We may guess these are cleaners and foragers.

4.2.2 Reconstruction

We expect the HRG model to include information other the hierarchical structure, such as average degree, average clustering and so on. So we may try to reconstruct the network based on the method we introduce before and compute these important information. We do 10 fits and compute their average degree and average clustering and take average.[5] The result is shown below:

We can see that it nicely estimate the average degree and average clustering in bee network, and in ant network, the average degree is very close to the real network, but the average clustering is not very close.

[5] In [11], dendrograms are generated from an equilibrium and a weight is each of them proportional to the square of their likelihood. But a sweep function seems not to be implemented in the igraph package in R. As simply doing several fits and taking the average still gives a nice result and this part is not the main topic of this paper, We will just use this method to display the information included in HRG method.

```
: print("recontructed average_degree=",average_rec("bee")[0])
  print("real average_degree=",average_degree(Ga1))
```

```
recontructed average_degree= 40.45575447570333
real average_degree= 40.3768115942029
```

```
: print("recontructed average_clustering=",average_rec("bee")[1])
  print("real average_clustering=",nx.average_clustering(Ga1))
```

```
recontructed average_clustering= 0.6713853711276926
real average_clustering= 0.678491112295436
```

```
: print("recontructed average_degree=",average_rec("ant")[0])
  print("real average_degree=",average_degree(Ants))
```

```
recontructed average_degree= 14.737895812053114
real average_degree= 14.584269662921349
```

```
: print("recontructed average_clustering=",average_rec("ant")[1])
  print("real average_clustering=",nx.average_clustering(Ants))
```

```
recontructed average_clustering= 0.3032390307468564
real average_clustering= 0.4250485186205191
```

5 Conclusion and Some Possible Further Direction

This project use several methods to predict the future links of bee network and ant network. Two static metrics AUC and precision are used to evaluate the effect of these methods. Most methods have a good AUC and precision values of them are not poor. Among these methods, degree product similarity has the highest AUC in bee network and AUC of LRW and SRW is the best in ant network. It is notable that LRW also performs well in bee network with respect to both AUC and precision. Common neighbour similarity has the highest precision in both networks. Besides of link prediction, HRG and SBM will provide more other information that reveals the structure of the networks, like community and hierarchical structure. But there are two questions, first is whether the AUC and precision can fairly evaluate the prediction effect. These two methods depend a lot on network structures, and it in some extent reveals how links are added to the network. But the AUC may work better with a stable network structure and varies with time. As it is difficult to know in what stage the network is at, it is hard to know how well the link prediction method works only through AUC. For the precision, The top size L is hard to be decided and it is sensitive to L. A minor change of L may cause the peak of the distribution of one similarity and thus reverse the ranking. In [17], more details are discussed and provide some sampling methods rather than randomly removing links. The problem of AUC and precision leads to a another question: is it possible to make us of time information, make prediction of future links with help of past information? We try to find the rule of links adding to networks at different time with certain time windows, and then fit the key parameters along the time and have a nicer prediction as the network evolves. It is reasonable to expect the network to grow as before and in fact, they also have very good AUC and precision scores. However, this method has a problem that the slope of curve at different time is hard to collect (We did it by hand) and the curve of slope is hard to fit. Maybe some machine learning method and better fitting model can improve the accuracy and convenience of this method. In addition, more research is done and we find a new area related to it, link predictions in temporal networks[16]. There are many more link predictions methods with concern of network evolution, like some new metrics including time information, tensor matrix factorization and a method also trying to use past information to predict future features[15]. Due to time limitation, I have not read these carefully, and more research related to this should be done later.

6 References

[1] Benjamin Blonder. *timeordered: Time-Ordered and Time-Aggregated Network Analyses*, 2018. R package version 0.9.9.

[2] Benjamin Blonder and Anna Dornhaus. Time-ordered networks reveal limitations to information flow in ant colonies. *PloS one*, 6(5):e20298, 2011.

[3] Aaron Clauset, Cristopher Moore, and M. E.J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191), 2008.

[4] Matthew J Hasenjager, William Hoppitt, and Ellouise Leadbeater. Network-based diffusion analysis reveals context-specific dominance of dance communication in foraging honeybees. *Nature communications*, 11(1):1–9, 2020.

[5] Hawoong Jeong, Zoltan Néda, and Albert-László Barabási. Measuring preferential attachment in evolving networks. *EPL (Europhysics Letters)*, 61(4):567, 2003.

[6] Brian Karrer and M. E.J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 83(1), 2011.

[7] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1), 1953.

[8] L. L. Linyuan and Tao Zhou. Link prediction in complex networks: A survey, 2011.

[9] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL*, 89(5), 2010.

[10] Danielle P. Mersch, Alessandro Crespi, and Laurent Keller. Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science*, 340(6136), 2013.

[11] M. E.J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 64(2 II), 2001.

[12] Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014.

[13] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014.

[14] Tiago P Peixoto. Nonparametric bayesian inference of the microcanonical stochastic block model. *Physical Review E*, 95(1):012317, 2017.

[15] Emile Richard, Nicolas Baskiotis, Theodoros Evgeniou, and Nicolas Vayatis. Link discovery using graph feature tracking. *Advances in Neural Information Processing Systems*, 23:1966–1974, 2010.

[16] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[17] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.