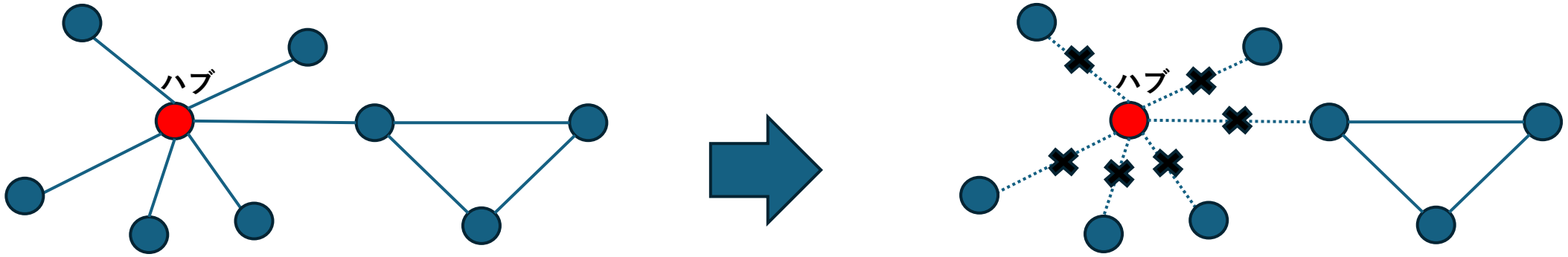
自己修復のデモ結果の確認と短い単純閉路の計算公式の導出

阿部伊吹

背景

ネットワークとは
・現実社会に多く存在し、ノード(点)とリンク(線)により構成→今回はこの共通の構造に着目
Ex).電力網、通信網、人間関係

ネットワークの特性
・スケールフリー性と呼ばれる構造を持つ
　極少数→多数のリンクを持つノード(ハブ)
　大多数→少数のリンクを持つノード
・ハブへの攻撃に非常に弱い



ハブへの攻撃後

→攻撃を受け、復元した際に、次の攻撃には耐えられる頑健性の高い自己修復が求められる

ラボローテーションでの目的
→自己修復デモを行い、ネットワークが攻撃を受け、修復した際の自己修復過程の確認を行う

実習内容(自己修復のデモ結果の確認)

実習内容

Network Graph Visualizerを用いて、様々なネットワークでの、攻撃率と修復率を変化させるデモを行う
→ノードの削除と、新リンクの追加過程見ることで、攻撃に対してのネットワークを修復過程を確認

1.BA_1_200_4_0モデル

Network Graph Visualizer

Settings

Network File

BA_1_200_4_0.net

Attack Method

Hub Attack

Graph Layout

Spring Layout

Attack Rate [%] 0 %

0% 50% 100%

Heal Rate [%] 0 %

0% 50% 100%

Draw Graph Refresh Graph

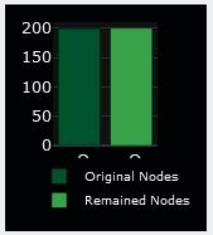
[Size Info.]

[Nodes]

Original Nodes: 200

Current Nodes: 200

Nodes Difference

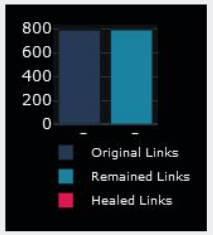


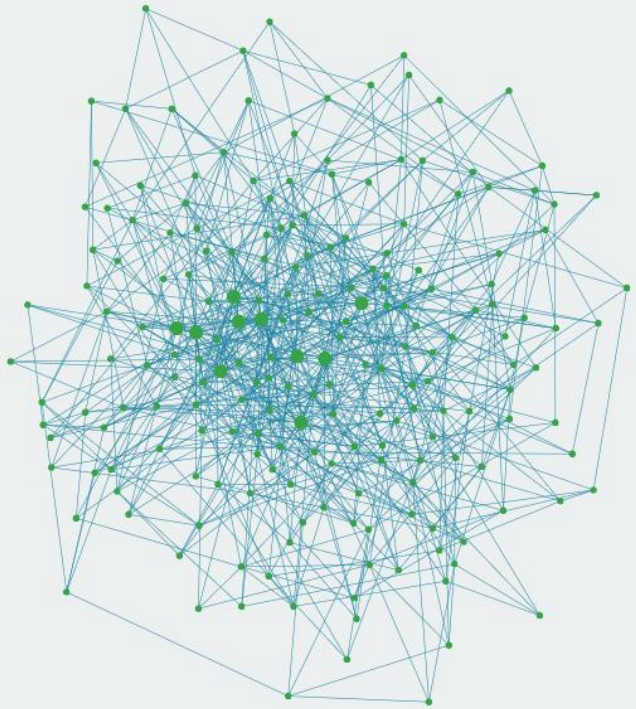
[Links]

Original Links: 790

Current Links: 790

Links Difference

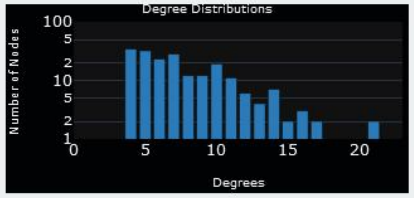




Degree Distribution: Original

Average Degree: 7.9

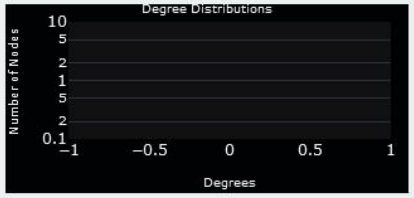
Maximum Degree: 22



Degree Distribution: Attaked

Average Degree:

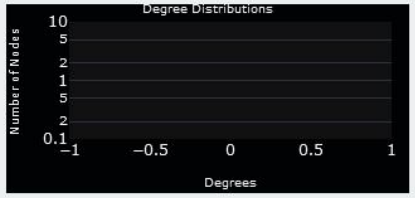
Maximum Degree:



Degree Distribution: Healed

Average Degree:

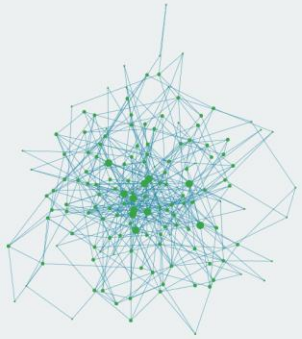
Maximum Degree:



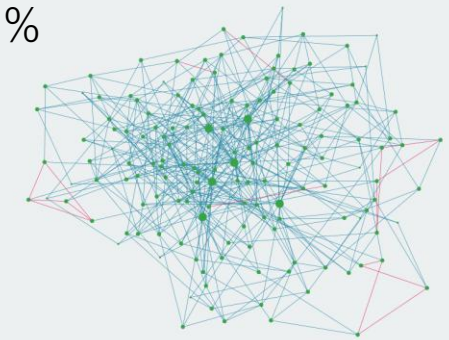
ds9.jaist.ac.jp:8050

攻撃率25%

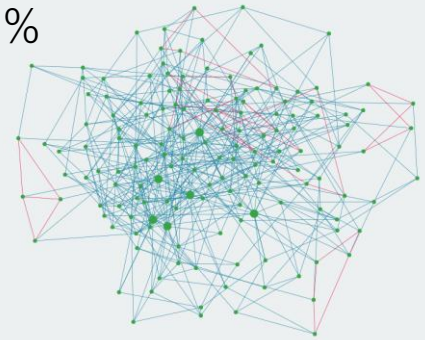
修復率0%



修復率5%



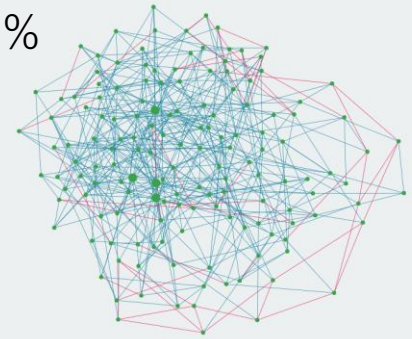
修復率10%

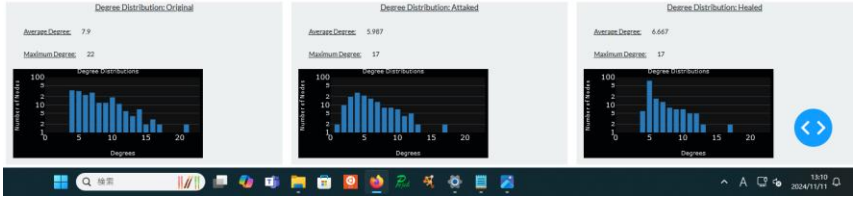


次数分布

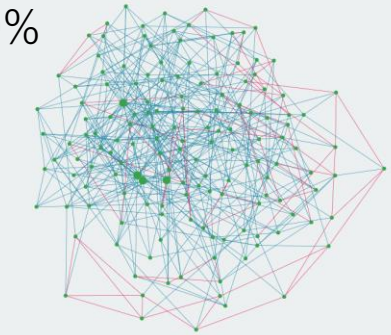
修復率を上げるにつれ、孤立したノードがないようにループ構造をとりながらリンクを繋げている

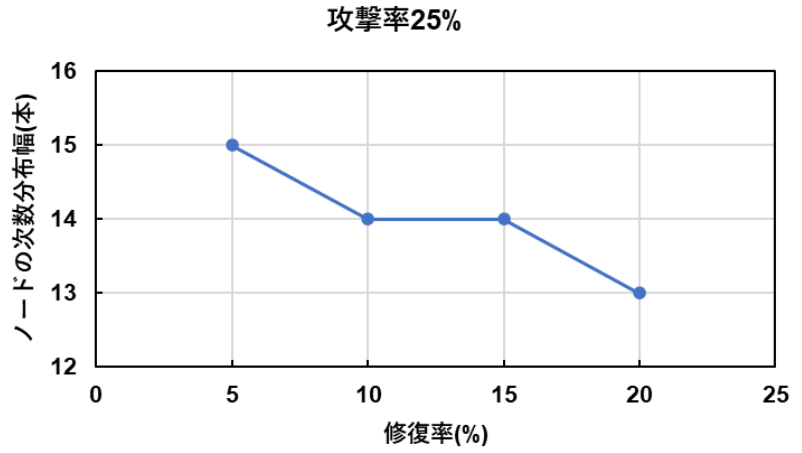
修復率15%





修復率20%

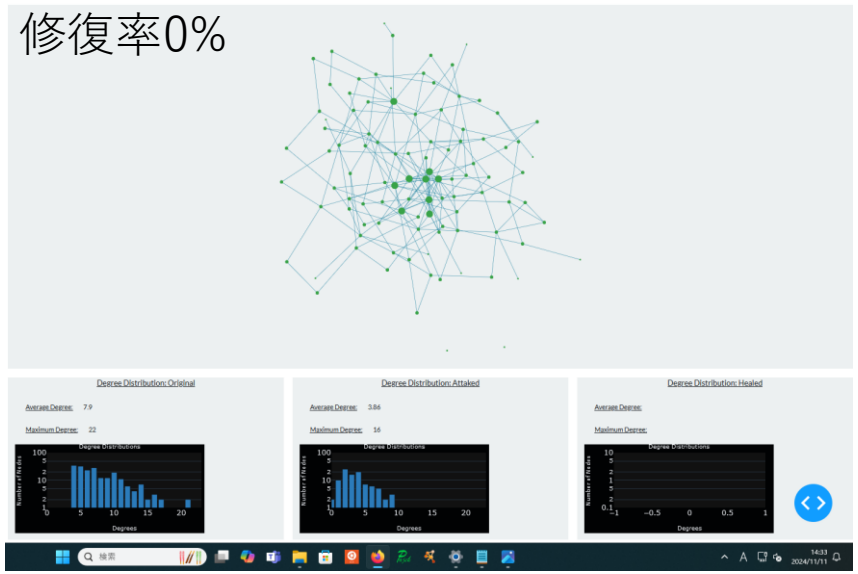




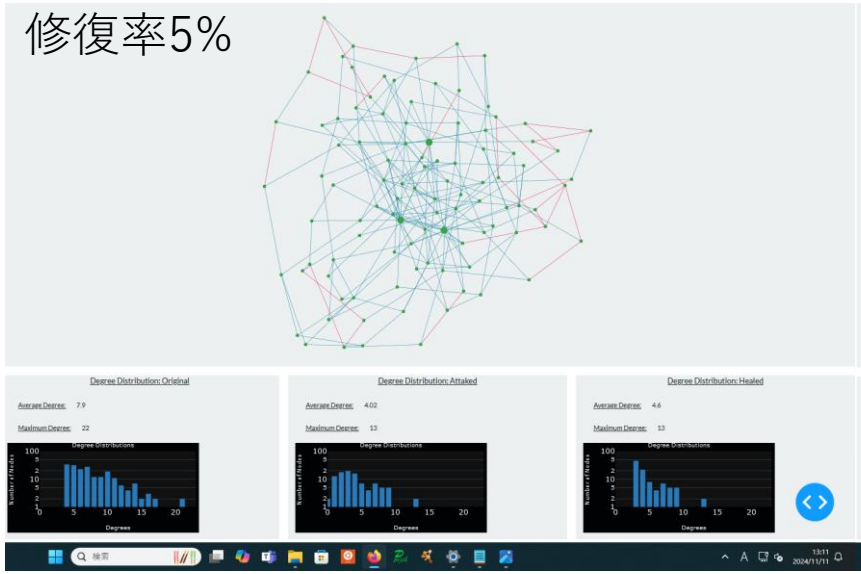
結果
修復率を上げるにつれ、ノードのリンクの次数分布幅が狭く
→リンクの公平化が行われている

攻撃率50%

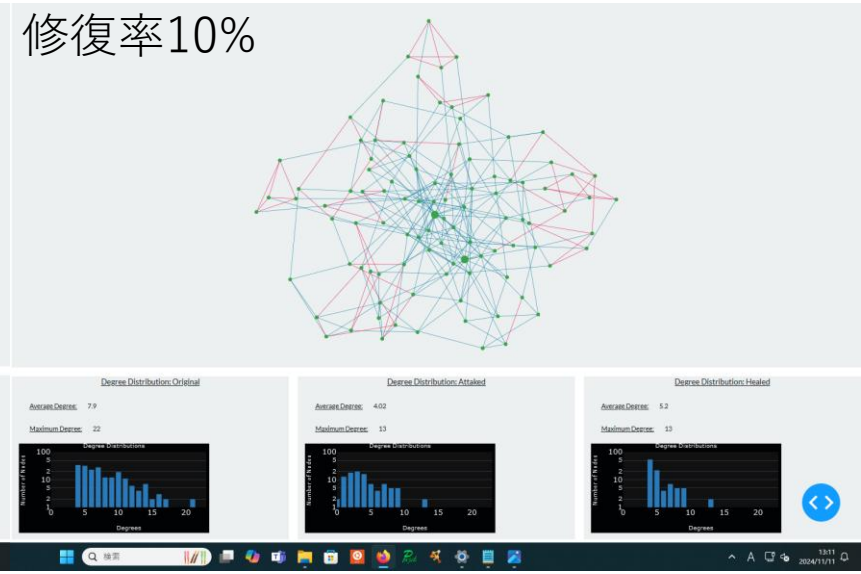
修復率0%



修復率5%

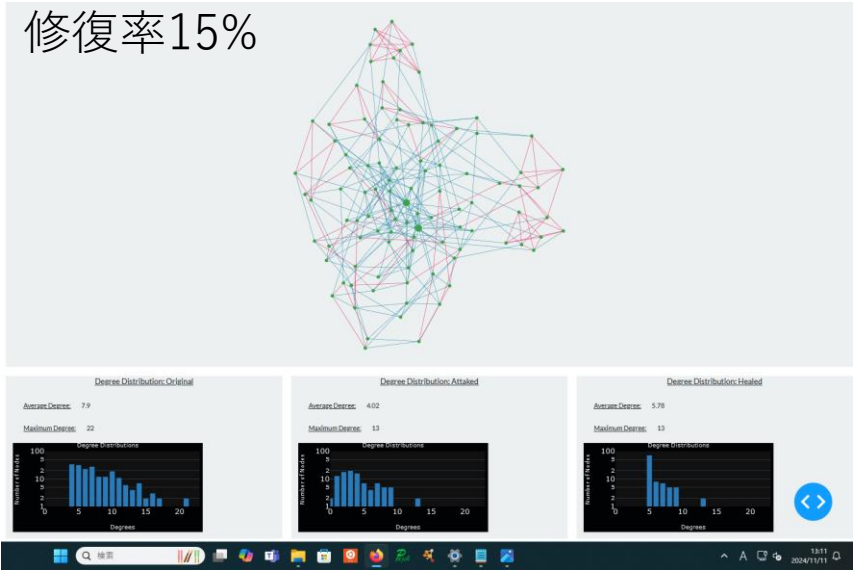


修復率10%

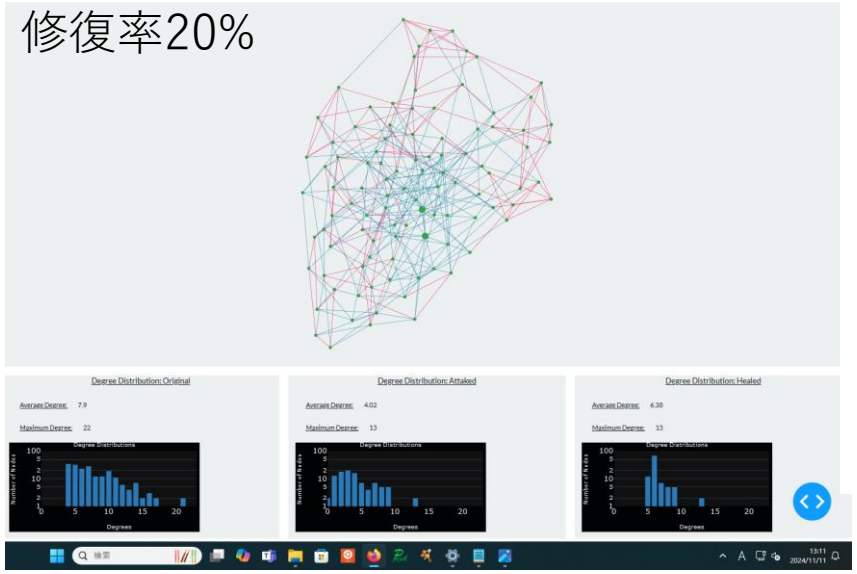


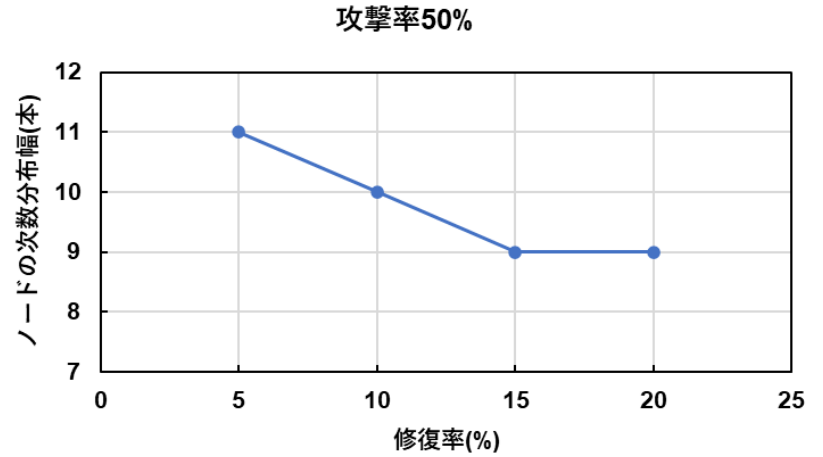
修復率を上げるにつれ、孤立したノードがないようにループ構造をとりながらリンクを繋げている

修復率15%



修復率20%

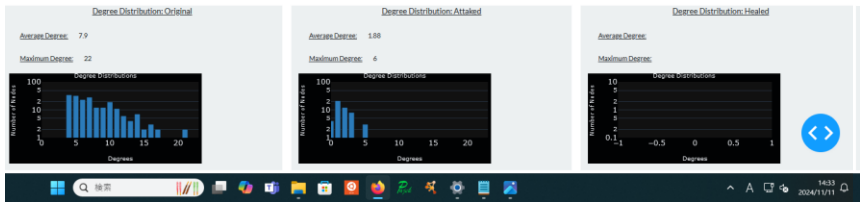




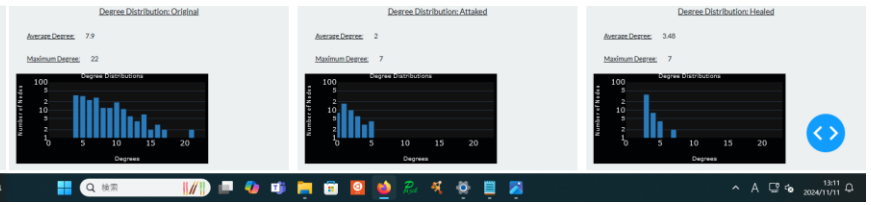
結果
修復率を上げるにつれ、ノードのリンクの次数分布幅が狭く
→リンクの公平化が行われている

攻撃率75%

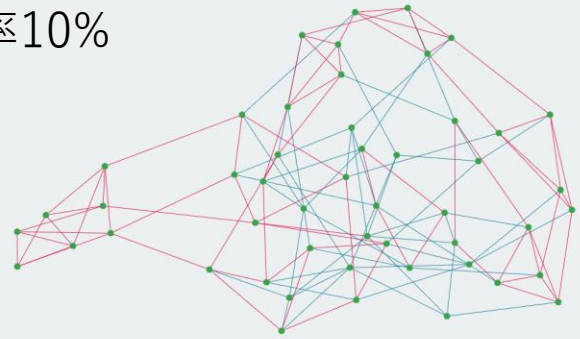
修復率0%



修復率5%



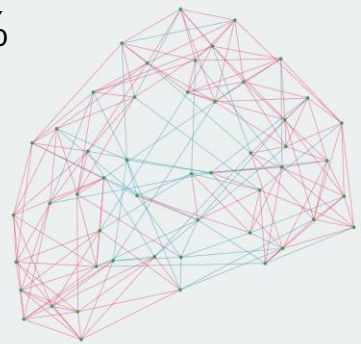
修復率10%

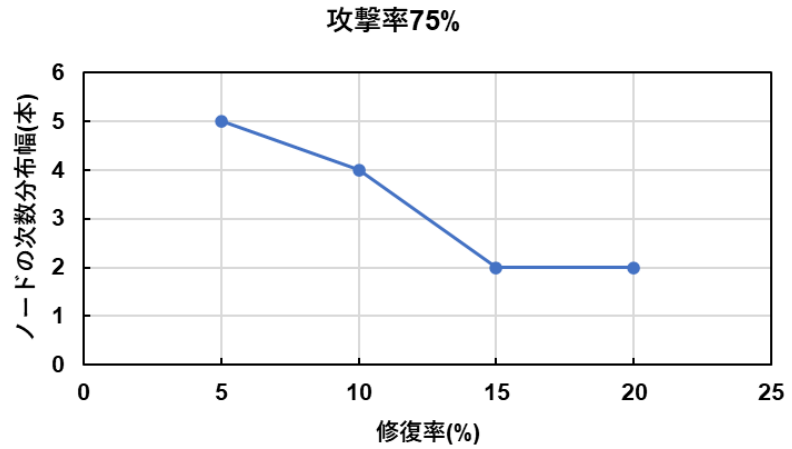


修復率を上げるにつれ、孤立したノードがないようにループ構造をとりながらリンクを繋げている

修復率15%

修復率20%





結果
修復率を上げるにつれ、ノードのリンクの次数分布幅が狭く
→リンクの公平化が行われている

2.Germany50

Network Graph Visualizer

Settings

Network File

Germany50.net

Attack Method

Hub Attack

Graph Layout

Original Layout

Attack Rate [%] 0 %

0% 50% 100%

Heal Rate [%] 0 %

0% 50% 100%

Draw Graph Refresh Graph

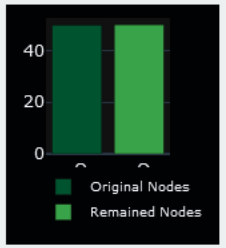
[Size Info.]

[Nodes]

Original Nodes: 50

Current Nodes: 50

Nodes Difference

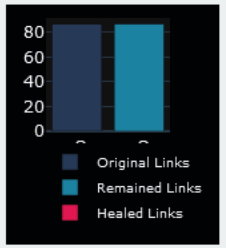


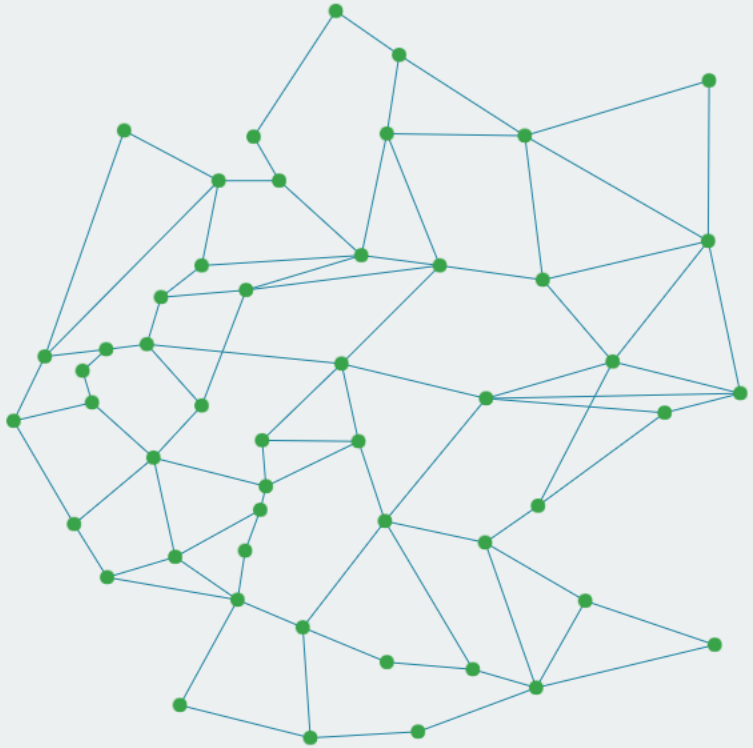
[Links]

Original Links: 87

Current Links: 87

Links Difference

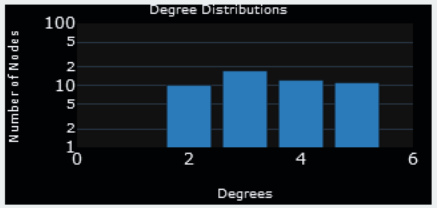




Degree Distribution: Original

Average Degree: 3.48

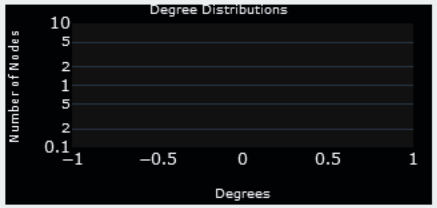
Maximum Degree: 5



Degree Distribution: Attaked

Average Degree:

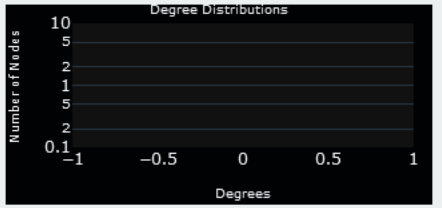
Maximum Degree:



Degree Distribution: Healed

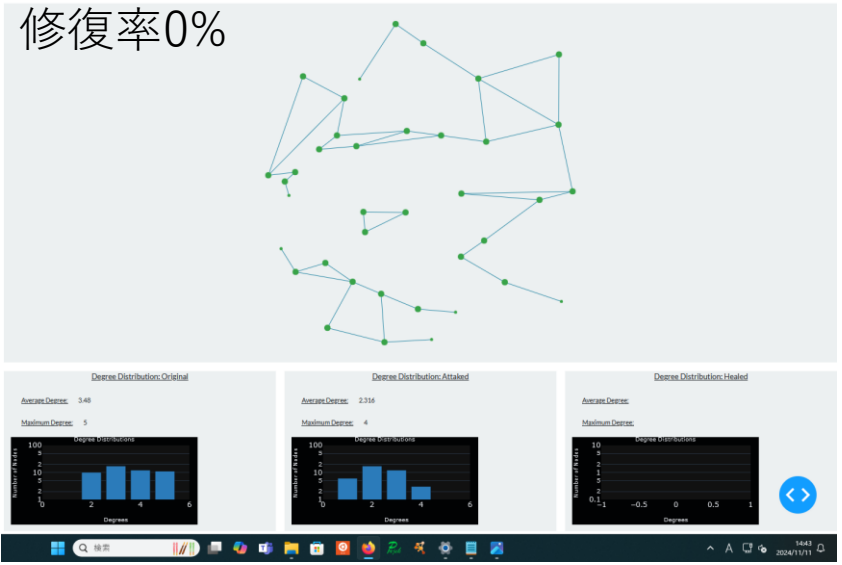
Average Degree:

Maximum Degree:

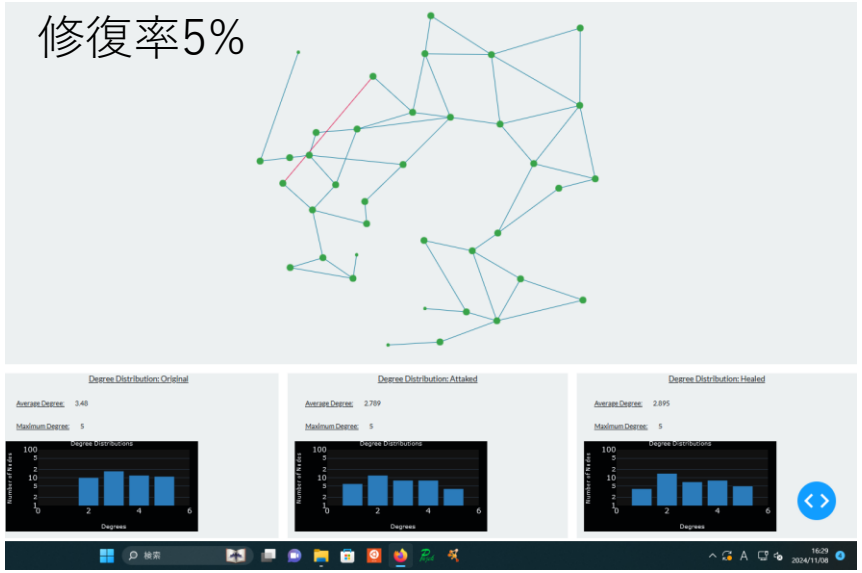


攻撃率25%

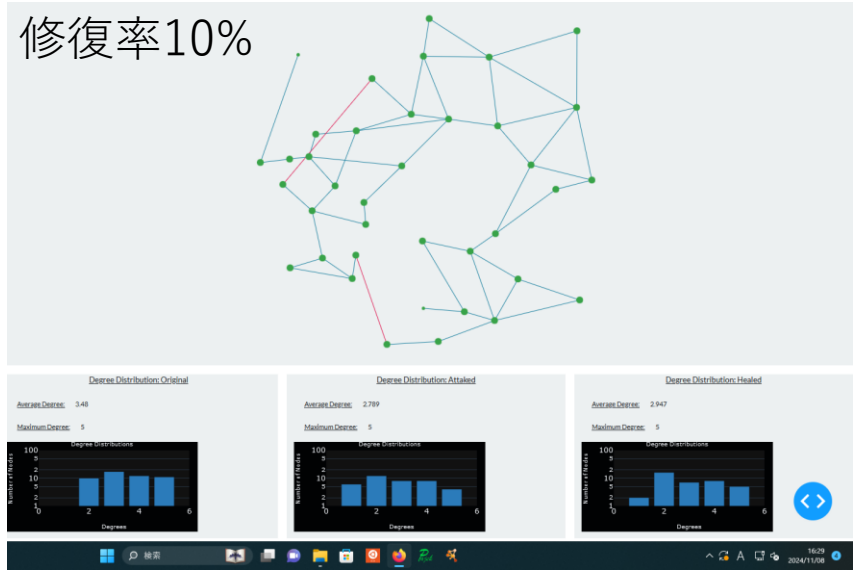
修復率0%



修復率5%

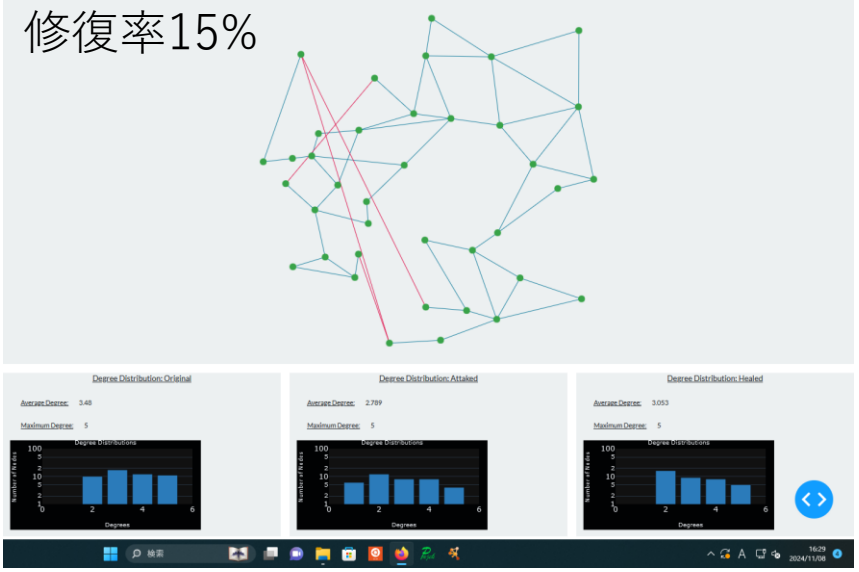


修復率10%

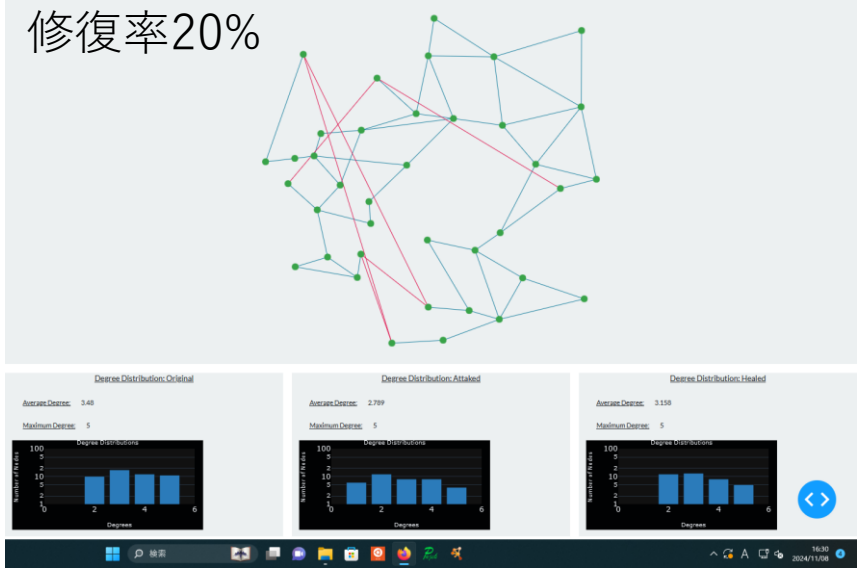


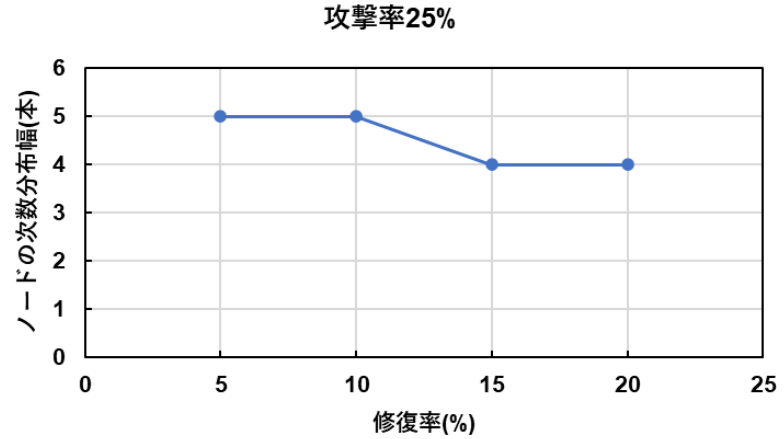
修復率を上げるにつれ、孤立したノードがないようにループ構造をとりながらリンクを繋げている

修復率15%



修復率20%

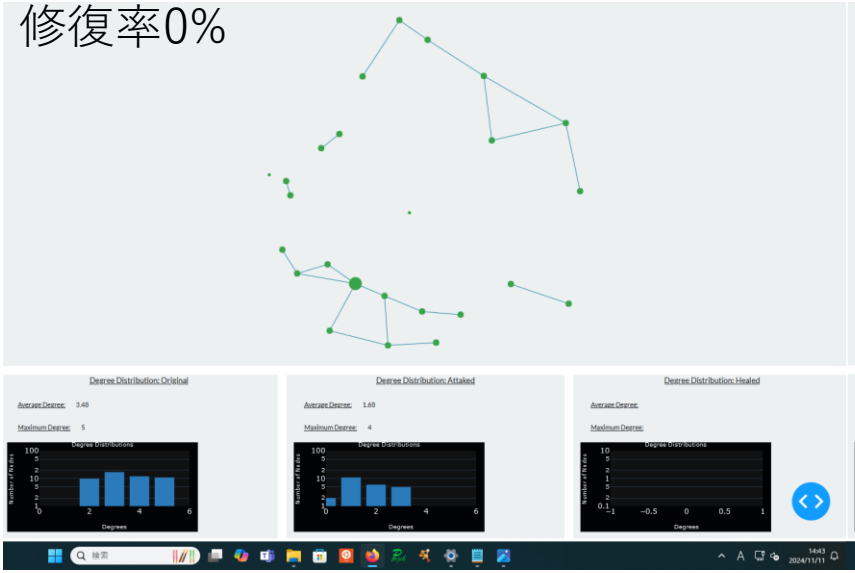




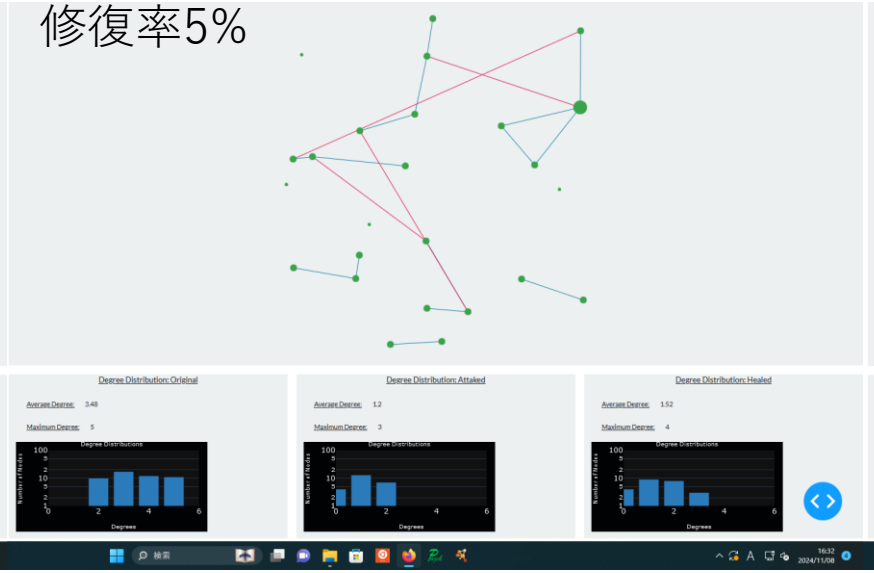
結果
修復率を上げるにつれ、ノードのリンクの次数分布幅が狭く
→リンクの公平化が行われている

攻撃率50%

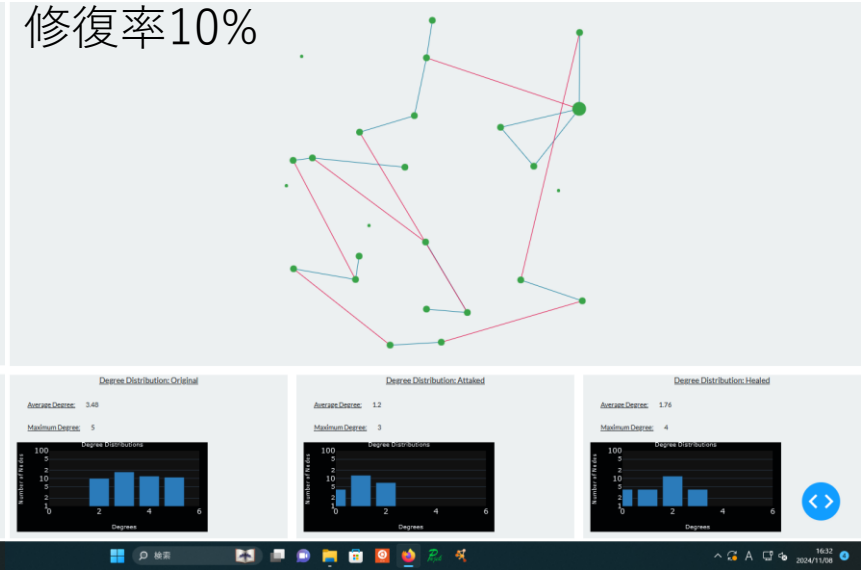
修復率0%

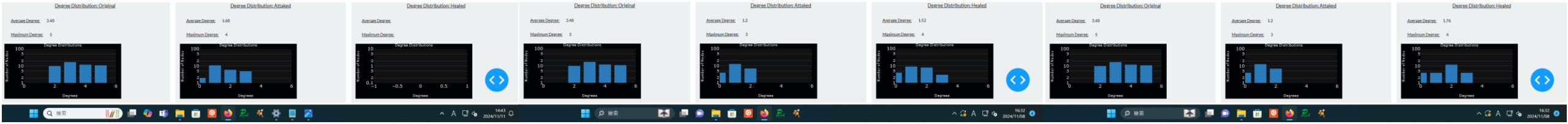


修復率5%



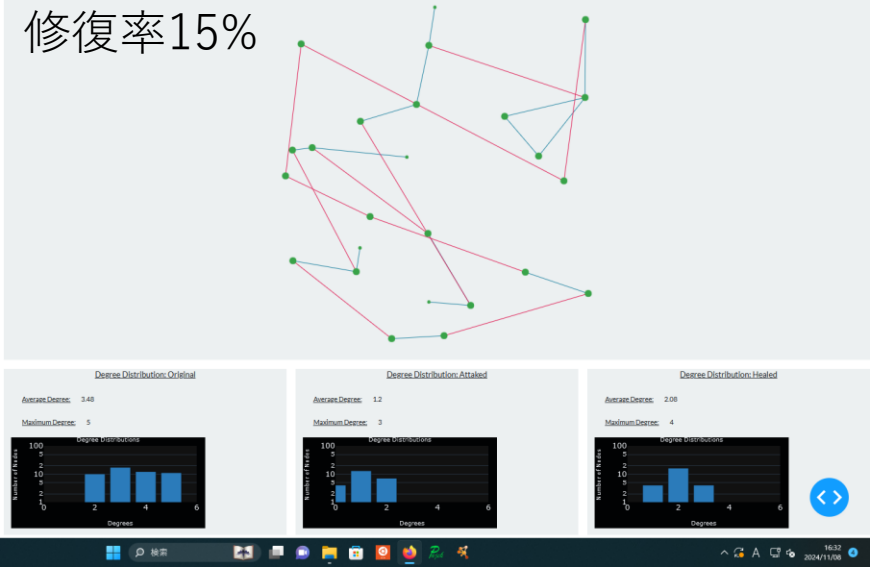
修復率10%



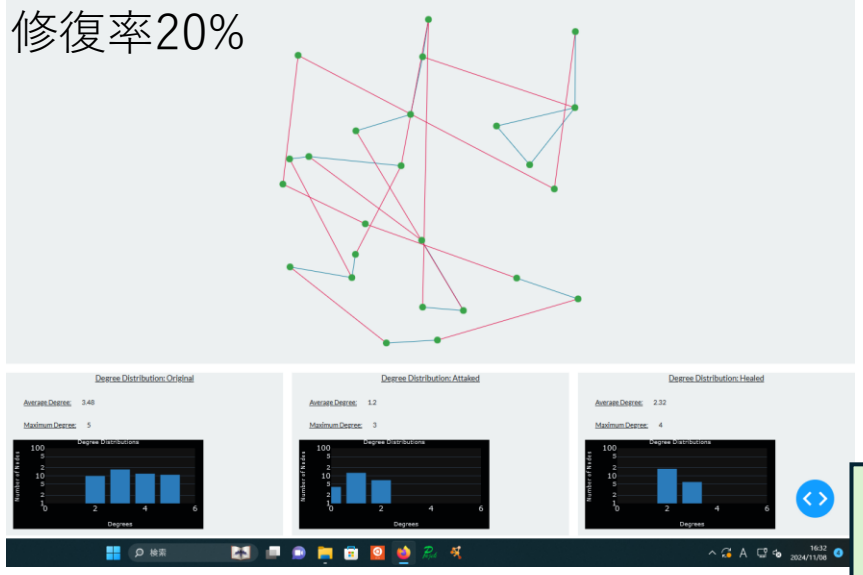


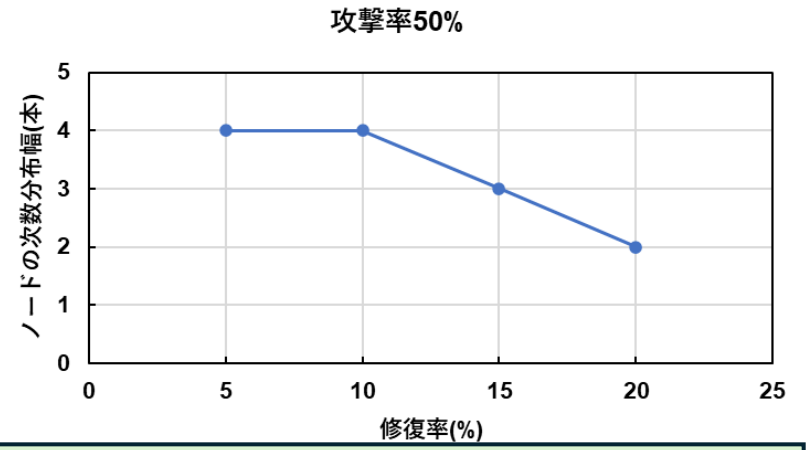
修復率を上げるにつれ、孤立したノードがないようにループ構造をとりながらリンクを繋げている

修復率15%



修復率20%

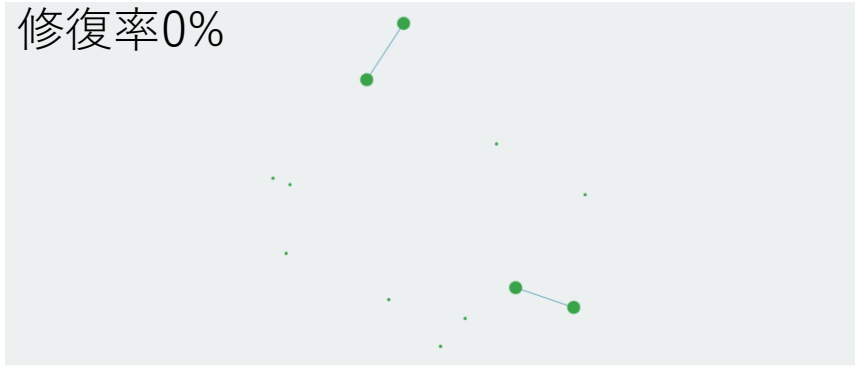


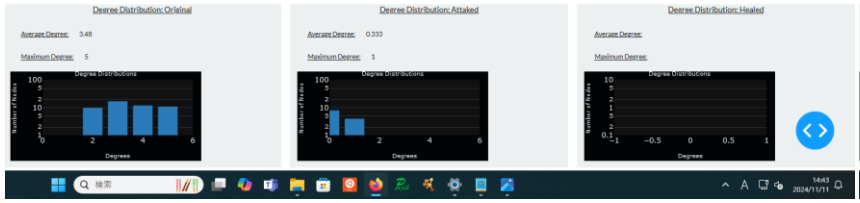


結果
修復率を上げるにつれ、ノードのリンクの次数分布幅が狭く
→リンクの公平化が行われている

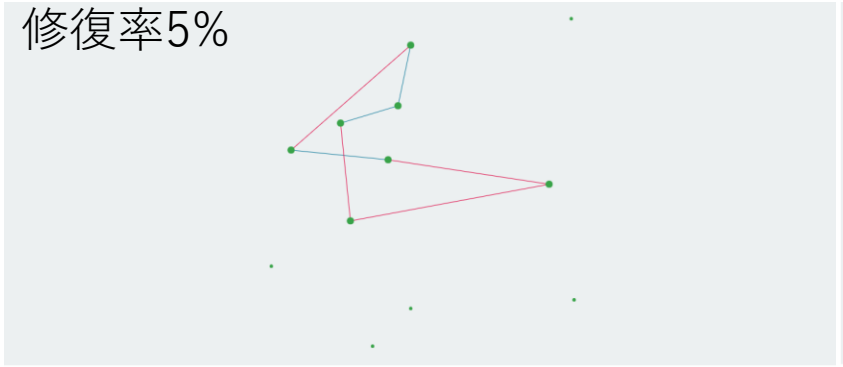
攻撃率75%

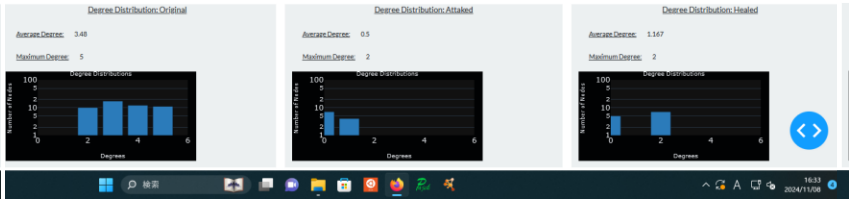
修復率0%



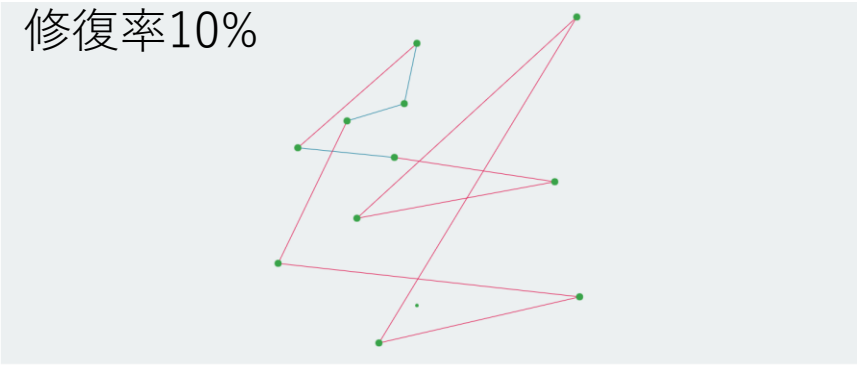


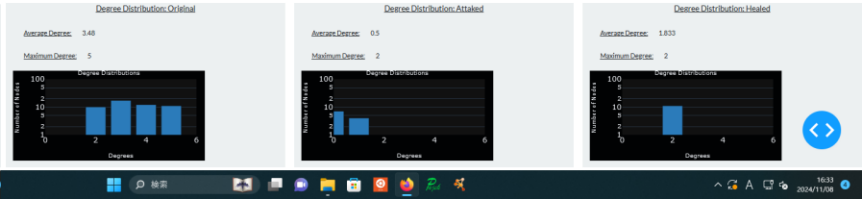
修復率5%





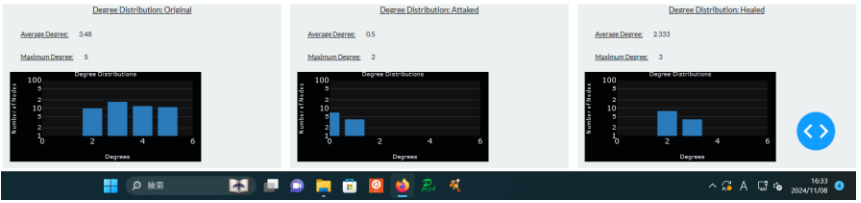
修復率10%



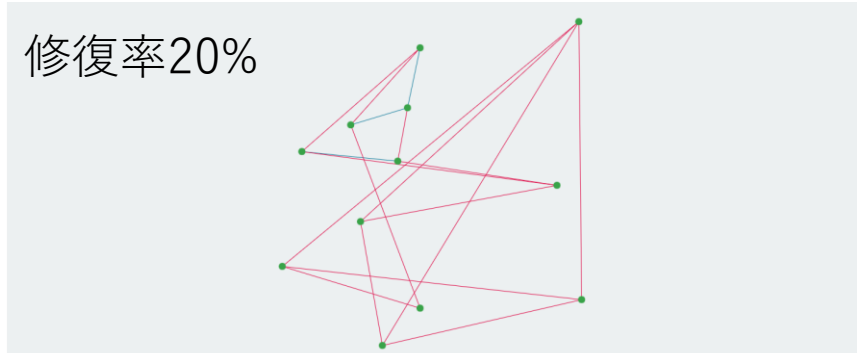


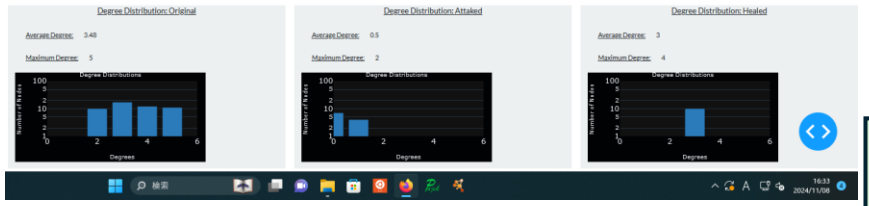
修復率を上げるにつれ、孤立したノードがないようにループ構造をとりながらリンクを繋げている

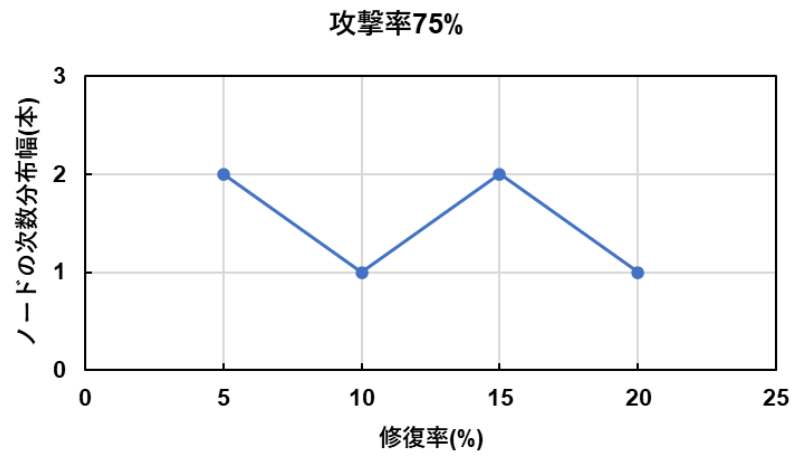
修復率15%



修復率20%





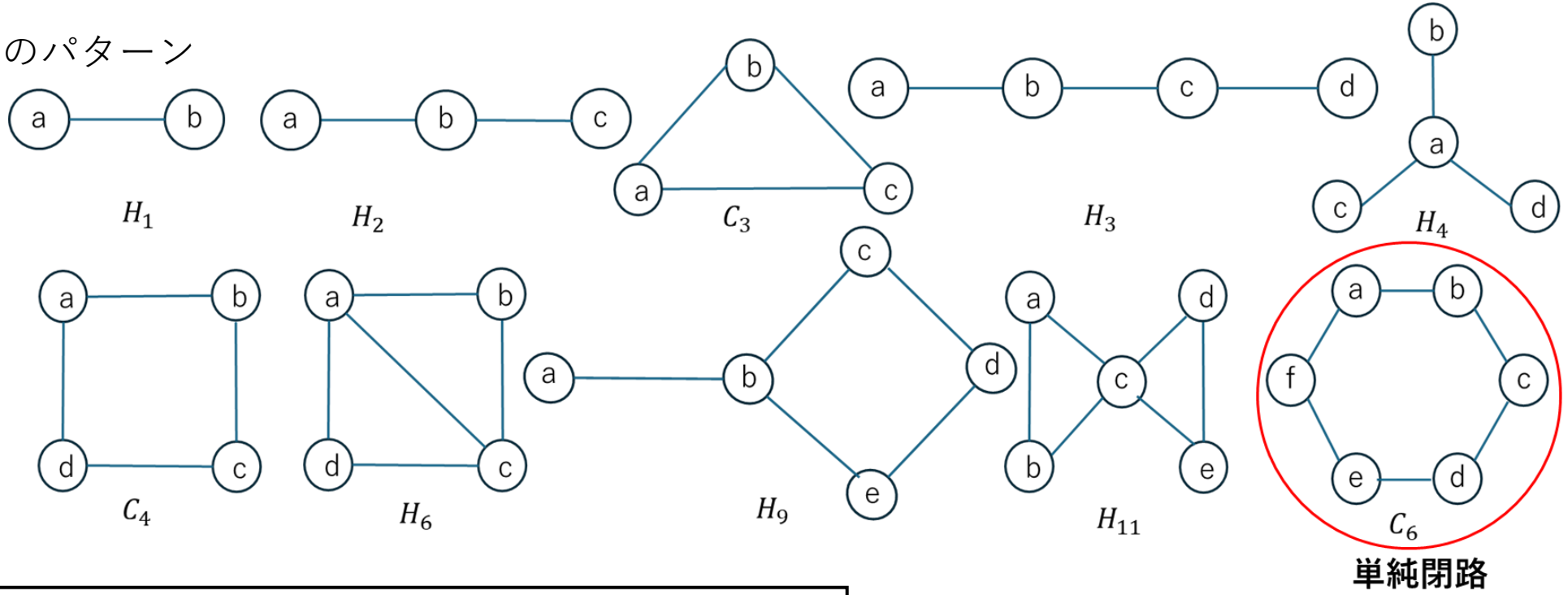


結果
ほぼノードが繋がっていないため、リンクの公平化を行いつつリンクを修復している

実習内容(短い単純閉路の計算公式の導出)

実習内容
長さl=6のサイクルの単純閉路(C6)を求めるときに、単純閉路以外のものを除く計算公式の導出
→具体的にはぞれぞれのパターン(H1～C6)に長さ 6 になる経路が何通りあるか確認し、その値を元に公式の導出

ℓ =6の時のパターン



ℓ サイクルの時の公式

$$n_G(C_l) = \frac{1}{2l}\left[Trace\left(A^l\right) - \sum_{H:N_{H<l}} C_l(H)\, n_G(H)\right]$$

ℓ =6の時の公式を導出する

H1の経路パターン

start	1	2	3	4	5	6
a	b	a	b	a	b	a
b	a	b	a	b	a	b

2通り

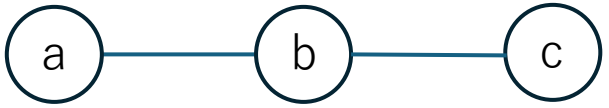
H2の経路パターン

start	1	2	3	4	5	6	
a	b	a	b	c	b	a	Aから始まるパターン(①往復)
a	b	c	b	c	b	a	Aから始まるパターン(②往復)
a	b	c	b	a	b	a	Aから始まるパターン(③往復)
c							Cから始まるパターン(①往復)**(Aと対称)**
c							**(Aと対称)**
c							**(Aと対称)**
b	a	b	a	b	c	b	b→aに行くパターン(①往復)
b	a	b	c	b	c	b	b→aに行くパターン(②往復)
b	a	b	c	b	a	b	
b	c	b	c	b	a	b	b→cに行くパターン(①往復)対称
b	c	b	a	b	a	b	
b	c	b	a	b	c	b	

a端から始める

c端から始める

bの真ん中から始める

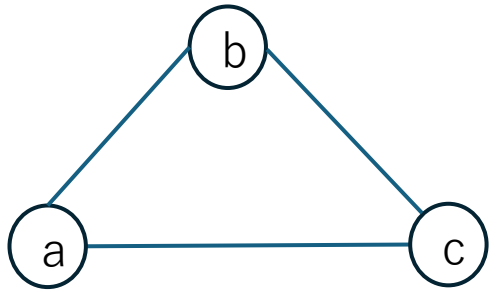


１２通り

C3の経路パターン

start	1	2	3	4	5	6	
a	b	c	a	b	c	a	右回り２週
a	c	b	a	c	b	a	左回り２週
a	b	c	a	c	b	a	右回り１週左周り１週
a	c	b	a	b	c	a	左回り１週右周り１週
a	b	a	c	b	c	a	
a	c	a	b	c	b	a	
a	b	c	b	a	c	a	
a	c	b	c	a	b	a	

a,b,cの３パターン



8×3=24通り

H3の経路パターン

start	1	2	3	4	5	6	
a	b	c	d	c	b	a	端から始める
d	c	b	a	b	c	d	端から始める(対称)
b	a	b	c	d	c	b	左に行く
b	c	d	c	b	a	b	右に行く
c							**bと対称**
c							**bと対称**

端から始める

真ん中から始める

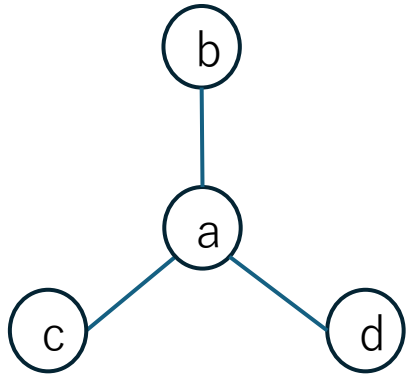
6通り

H4の経路パターン

start	1	2	3	4	5	6	
a	b	a	c	a	d	a	aから始めるパターン1番目ｂ右周り
a	b	a	d	a	c	a	aから始めるパターン1番目ｂ左周り
a	c						aから始めるパターン1番目ｃ右周り
a	c						
a	d						aから始めるパターン1番目c右周り
a	d						
b	a	c	a	d	a	b	bから始めるパターン右周り
b	a	d	a	c	a	b	bから始めるパターン左周り
c	a	d	a	b	a	c	cから始めるパターン右周り
c	a	b	a	d	a	c	cから始めるパターン左周り
d	a						dから始めるパターン右周り
d							dから始めるパターン左周り

中心から始める

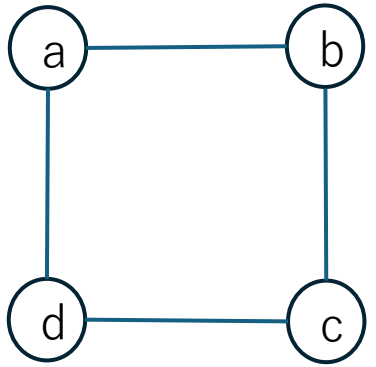
端の点から始める



12通り

C4の経路パターン

start	1	2	3	4	5	6	
a	b	a	b	c	d	a	右周り最初に戻る
a	b	c	b	c	d	a	右周り2番目に戻る
a	b	c	d	c	d	a	右周り３番目に戻る
a	b	c	d	a	d	a	右周り最後に戻る
a	d	a	d	c	b	a	左周り最初に戻る
a	d	c	d	c	b	a	左周り２番目に戻る
a	d	c	b	c	b	a	左周り３番目に戻る
a	d	c	b	a	b	a	左周り最後に戻る
a	b	a	d	c	b	a	最初行って反対方向
a	d	a	b	c	d	a	最初行って反対方向の逆回り



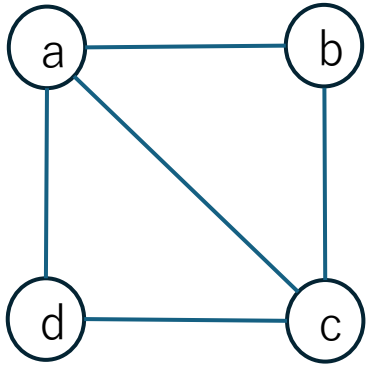
各点のa,b,c,dについて考える

１０×４=40通り

H6

start	1	2	3	4	5	6	
a	b	c	a	c	d	a	上の三角形から作る
a	b	c	a	d	c	a	上の三角形から作る
a	c	b	a	c	d	a	上の三角形から作る
a	c	b	a	d	c	a	上の三角形から作る
a	c	b	a	b	c	a	下の三角形から作る
a	c	b	a	c	d	a	下の三角形から作る
a	d	c	a	b	c	a	下の三角形から作る
a	d	c	a	c	b	a	下の三角形から作る
a	c	b	a	d	c	a	a→cから一周
a	c	d	a	b	c	a	
a	b	c	d	a	c	a	一周してからa→c
a	d	c	b	a	c	a	一周してからa→c
d	a	b	c	a	c	d	一周して最後スラッシュ
d	c	b	a	c	a	d	一周して最後スラッシュ
d	a	c	a	b	c	d	最初スラッシュして一周
d	c	a	c	b	a	d	
d	a	c	b	a	c	d	クロスして一周
d	c	a	b	c	a	d	

aから始める

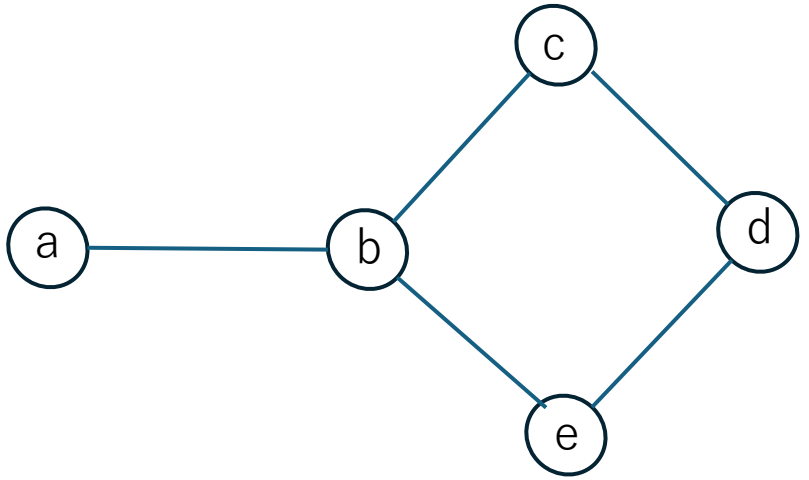


bから始める

18×２＝36通り

H9の経路パターン

start	1	2	3	4	5	6	
a	b	c	d	e	b	a	Aから始める右周り
a	b	e	d	c	b	a	Aから始める左周り
b	a	b	c	d	e	b	b→aから始める右周り
b	a	b	e	d	c	b	b→aから始める左周り
b	c	d	e	b	a	b	一周回ってからb→a右周り
b	e	d	c	b	a	b	一周回ってからb→a左周り
c	d	e	b	a	b	c	Cから始める右回り
c	b	a	b	e	d	c	Cから始める左回り
e	b	a	b	c	d	e	eから始める右回り
e	d	c	b	a	b	e	eから始める左回り
d	e	b	a	b	c	d	dから始める右回り
d	c	b	a	b	e	d	dから始める左回り

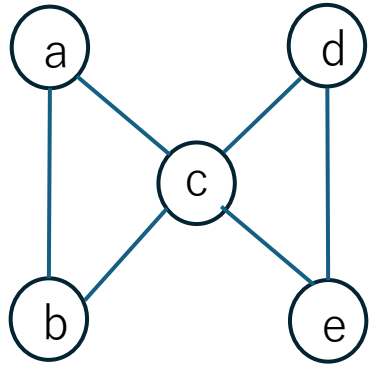


12通り

H11

start	1	2	3	4	5	6	
a	b	c	e	d	c	a	bからの外周
a	c	d	e	c	b	a	Cからの外周
a	b	c	d	e	c	a	Bからのクロス
a	c	e	d	c	b	a	Cからのクロス
c	a	b	c	e	d	c	aからの外周
c	b	a	c	d	e	c	bからの外周
c	a	b	v	d	e	c	aからのクロス
c	b	a	c	e	d	c	bからのクロス
c	d	e	c	b	a	c	dからの外周
c	e	d	c	a	b	c	eからの外周
c	d	e	c	a	b	c	dからのクロス
c	e	d	c	b	a	c	eからのクロス

頂点a,b,e,d,の４通り



４×４+8=24通り

計算公式の導出

ℓ サイクルの時の公式

$$n_G(C_l) = \frac{1}{2l}\left[Trace\left(A^l\right) - \sum_{H:N_{H<l}} C_l(H)\, n_G(H)\right]$$

ℓ =6の時の公式を導出する

ℓ =6の時、求めた係数を代入

$$\boldsymbol{n_G(C_6) = \frac{1}{12}\left[Trace\left(A^6\right) - 2n_G(H_1) - 12n_G(H_2) - 24n_G(C_3) - 6n_G(H_3) - 12n_G(H_4) - 40n_G(C_4) - 36n_G(H_6) - 12n_G(H_9) - 24n_G(H_{11})\right]}$$

結果

・単純閉路の計算公式の導出では公式を求める過程を学習することで、ネットワーク解析でのサイクル検出のアルゴリズムを学習することができた