SIECI WIELOWARSTWOWE

Karol Działowski
Zachodniopomorski Uniwersytet Technologiczny

16 listopada 2020

Spis treści

1 Wstęp

Celem laboratorium nr. 6 było zapoznanie się z podstawowymi funkcjami do przetwarzania sieci wielowarstwowych. Sieć wielowarstwowa to sieć składająca się z wielu sieci klasycznych budujących tak zwane warstwy. Takie rodzaje sieci mogą służyć na przykład do przedstawiania relacji społecznych, połączenia lotnicze itd.

2 Zapoznanie się z podstawowymi funkcjami do przetwarzania sieci wielowarstwowych

Zapoznano się z poniższymi działaniami/funkcjami:

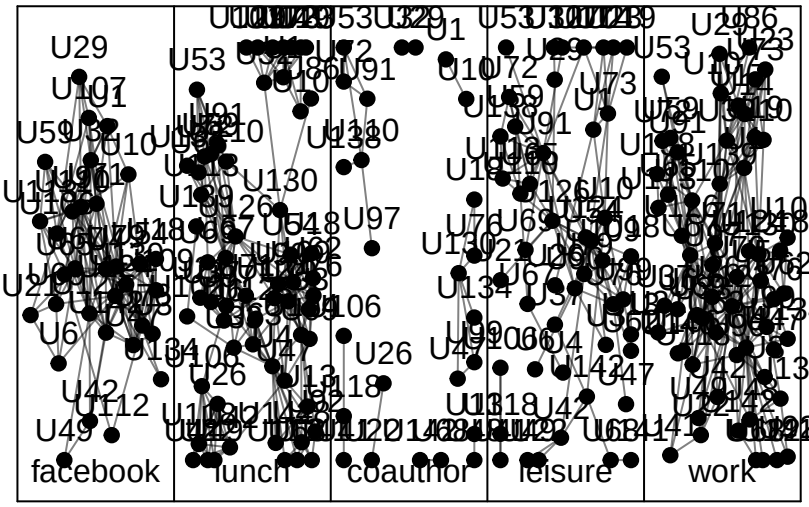
- instalacja pakietu *multinet*
- podstawowe miary dla sieci wielowarstwowych
- zależności międzywarstwowe
- sąsiedztwo w warstwach
- sieci prefefiniowane
- podobieństwo warstw
- odczyt i zapis sieci

2.1 Instalacja pakietu *multinet*

```
install.packages("multinet")
library(multinet)
library(igraph)
```

2.2 Podstawowe miary dla sieci wielowarstwowych

```
> net <- ml_aucs()
> net
Multilayer Network [61 actors, 5 layers, 224 vertices, 620 edges (620,0)]
> plot(net)
> # degrees of all actors, considering edges on all layers
> degree_ml(net)
 [1] 49 31 18 22  9 18 16 13 16 17 15  9 11 32 24 13 19 25 13 22 44 35 12 25 10  7 20 23 46 44
[31] 21 30 16 32 25 11 18 17 18 17 19 41  2  7 20 19 23 19 11 47  3 17 11 12  2 12  7 21 27 18
[61] 39
> # degree of actors U54 and U3, only considering layers work and coauthor
> degree_ml(net,c("U54","U3"),c("work","coauthor"),"in")
[1] 8 2
> # an indication of whether U54 and U3 are selectively active only on some layers
> degree_deviation_ml(net,c("U54","U3"))
[1] 3.577709 2.870540
> # co-workers of U3
> neighborhood_ml(net,"U3","work")
[1] 2
> # co-workers of U3 who are not connected to U3 on other layers
> xneighborhood_ml(net,"U3","work")
[1] 1
> # percentage of neighbors of U3 who are also co-workers
> relevance_ml(net,"U3","work")
[1] 0.1818182
> # redundancy between work and lunch
> connective_redundancy_ml(net,"U3",c("work","lunch"))
[1] 0.1111111
> # percentage of neighbors of U3 who would no longer
> # be neighbors by removing this layer
> xrelevance_ml(net,"U3","work")
[1] 0.09090909
```



Rysunek 1: AUCS network

2.3 Atrybuty

```
> # setting attributes based on network properties: create a "degree"
> # attribute and set its value to the degree of each actor
> actors_ml(net) -> a
> layers_ml(net) -> l
> degree_ml(net,actors=a,layers=l,mode="all") -> d
> add_attributes_ml(net,target="actor",type="numeric",attributes="degree")
> set_values_ml(net,attribute="degree",actors=a,values=d)
> get_values_ml(net,attribute="degree",actors="U54")
  value
1    35
> # select actors based on attribute values (e.g., with degree greater than 40)
> get_values_ml(net,attribute="degree",actors=a) -> degrees
> a[degrees>40]
[1] "U4"   "U123" "U91"  "U79"  "U110" "U67"
> # list all the attributes
> attributes_ml(net)
    name   type
1  group string
2   role string
3 degree double
```

2.4 Algorytmy detekcji społeczności

```
> # Algorytmy detekcji spolecznosci
```

```
> net <- ml_aucs()
> abacus_ml(net)
Warning: could not run external library: File not found: Cannot open input tmp file
Returning empty community set.
[1] actor layer cid
<0 wierszy> (lub 'row.names' o zerowej dugoci)
> clique_percolation_ml(net)
    actor    layer cid
1      U4    lunch   0
2      U4     work   0
...
332  U109    lunch  22
333  U109  leisure  22
 [ reached 'max' / getOption("max.print") -- omitted 175 rows ]
> glouvain_ml(net)
    actor    layer cid
1      U1    lunch   0
2      U1 coauthor   0
...
223  U126  leisure   5
224  U126     work   5
```

2.5 Konwersja do sieci single layer

```
> net <- ml_aucs()
> # using the default merge.actors=TRUE we create a multigraph,
> # where each actor corresponds to a vertex in the result
> multigraph <- as.igraph(net)
> # this is a simple graph corresponding to the facebook layer
> facebook1 <- as.igraph(net, "facebook")
> # this includes also the actors without a facebook account
> facebook2 <- as.igraph(net, "facebook", all.actors=TRUE)
> # two layers are converted to an igraph object, where two
> # vertices are used for each actor: one corresponding to the
> # vertex on facebook, one to the vertex on lunch
> f_l_net <- as.igraph(net, c("facebook","lunch"),merge.actors=FALSE)
```

2.6 Odległości w sieciach multilayer

```
> distance_ml(net, from='U54', to=character(0), method="multiplex")
    from   to facebook lunch coauthor leisure work
1    U54 U142        1     0        0       0    0
2    U54 U142        0     0        0       2    0
...
141  U54 U102        0     2        1       2    0
142  U54 U102        0     2        0       3    0
 [ reached 'max' / getOption("max.print") -- omitted 554 rows ]
> net <- ml_aucs()
> distance_ml(net,"U54","U3")
  from to coauthor lunch facebook leisure work
1  U54 U3        0     1        0       0    0
2  U54 U3        0     0        1       0    0
3  U54 U3        0     0        0       1    0
4  U54 U3        0     0        0       0    2
```

2.7 Zapis/odczyt sieci z pliku

```
> file <- tempfile("aucs.mpx") # zapis do R multinet/extdata
> net <- ml_aucs()
> write_ml(net,file)
> net1 <- read_ml(file,"AUCS")
```

```
> net1
Multilayer Network [61 actors, 5 layers, 224 vertices, 620 edges (620,0)]
```

2.8 Tworzenie sieci wielowarstwowej

```
> # Tworzenie sieci wielowarstwowej
> net <- ml_empty()
> # Adding some layers
> add_layers_ml(net,"l1")
> add_layers_ml(net,c("l2","l3"),c(TRUE,FALSE))
> layers_ml(net)
[1] "l3" "l2" "l1"
> # Adding some vertices (actor A3 is not present in layer l3: no corresponding vertex there)
> vertices <- data.frame(
+   c("A1","A2","A3","A1","A2","A3"),
+   c("l1","l1","l1","l2","l2","l2"))
> add_vertices_ml(net,vertices)
> vertices <- data.frame(
+   c("A1","A2"),
+   c("l3","l3"))
> add_vertices_ml(net,vertices)
> vertices_ml(net)
  actor layer
1    A2    l3
2    A1    l3
3    A2    l2
4    A1    l2
5    A3    l2
6    A2    l1
7    A1    l1
8    A3    l1
```

2.9 Edycja istniejącej sieci

```
> # Verifying that the actors have been added correctly
> num_actors_ml(net)
[1] 3
> actors_ml(net)
[1] "A2" "A1" "A3"
> # We create a data frame specifying two edges:
> # A2,l2 -- A3,l1
> # A2,l2 -- A3,l2
> edges <- data.frame(
+   c("A2","A2"),
+   c("l2","l2"),
+   c("A3","A3"),
+   c("l1","l2"))
> add_edges_ml(net,edges)
> edges_ml(net)
  from_actor from_layer to_actor to_layer dir
1         A2         l2       A3       l2   1
2         A2         l2       A3       l1   0
> # The following deletes layer 1, and also deletes
> # all vertices from "l1" and the edge with an end-point in "l1"
> delete_layers_ml(net,"l1")
> # The following also deletes the vertices associated to
> # "A1" in layers "l2" and "l3"
> delete_actors_ml(net,"A1")
> # deleting vertex A2,l3 and edge A2,l2 -- A3,l2
> delete_vertices_ml(net,data.frame("A2","l3"))
> edges <- data.frame("A2","l2","A3","l2")
> delete_edges_ml(net,edges)
```

```
> net
Multilayer Network [2 actors, 2 layers, 2 vertices, 1 edge (0,1)]
```

2.10 Tworzenie sieci ewoluującej

```
> # we generate a network with two layers, one growing according
> # to the Preferential Attachment model and one growing by selecting
> # new edges uniformly at random.
> models <- c(evolution_pa_ml(3,1), evolution_er_ml(50))
> # all the probability vectors must have the same number of
> # fields, one for each layer: two in this example
> # by defining pr.internal and pr.external, we are also implicitely defining
> # pr.no.action (1 minus the other probabilities, for each field/layer).
> pr_external <- c(.5,0)
> pr_internal <- c(.5,.5)
> # each layer will import edges from the other if needed
> # (not the second layer in this example: it has 0 probability of external events)
> dependency <- matrix(c(0,1,1,0),2,2)
> # 100 steps of network growing, adding actors from a pool of 100
> grow_ml(100, 100, models, pr_internal, pr_external, dependency)
Multilayer Network [100 actors, 2 layers, 150 vertices, 122 edges (122,0)]
```

2.11 Porównywanie warstw

```
> layer_comparison_ml(net,method="kulczynski2.actors")
    l3  l2
l3 NaN NaN
l2 NaN   1
> layer_comparison_ml(net,method="kulczynski2.edges")
    l3  l2
l3 NaN NaN
l2 NaN NaN
> layer_comparison_ml(net,method="kulczynski2.triangles")
    l3  l2
l3 NaN NaN
l2 NaN NaN
> layer_comparison_ml(net,method="hamann.actors")
   l3 l2
l3  1 -1
l2 -1  1
> layer_comparison_ml(net,method="hamann.edges")
   l3 l2
l3  1  1
l2  1  1
> layer_comparison_ml(net,method="hamann.triangles")
    l3  l2
l3 NaN NaN
l2 NaN NaN
> # comparison of degree distributions (divergences)
> layer_comparison_ml(net,method="dissimilarity.degree")
    l3  l2
l3 NaN NaN
l2 NaN   0
> layer_comparison_ml(net,method="KL.degree")
           l3        l2
l3  0.0000000 0.6931472
l2 -0.3465736 0.0000000
> layer_comparison_ml(net,method="jeffrey.degree")
    l3 l2
l3 NaN  0
l2   0  0
> # statistical degree correlation
```

```
> layer_comparison_ml(net,method="pearson.degree")
    l3  l2
l3 NaN NaN
l2 NaN NaN
> layer_comparison_ml(net,method="rho.degree")
    l3  l2
l3 NaN NaN
l2 NaN NaN
```

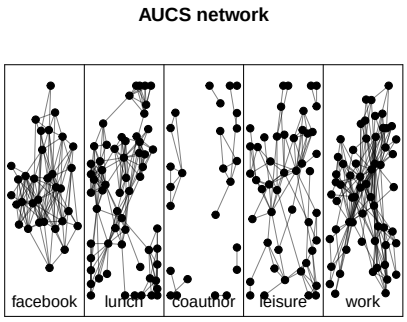
2.12 Wizualizacja

```
# Slajd 19
# Wizualizacja
net <- ml_florentine()
layout_multiforce_ml(net, w_in = 1, w_inter = 1, gravity = 0, iterations = 100)
layout_circular_ml(net)
net <- ml_florentine()
layout_multiforce_ml(net)
l <- layout_circular_ml(net)
plot(net,layout=l)
net <- ml_aucs()
layout_multiforce_ml(net)
l <- layout_circular_ml(net)
plot(net,layout=l)
net <- ml_florentine()
plot(net)
c <- clique_percolation_ml(net)
plot(net, vertex.labels.cex=.5, com=c)
net <- ml_aucs()
plot(net, vertex.labels=NA)
title("AUCS network")
values2graphics(c("a", "b", "b", "c"))
```

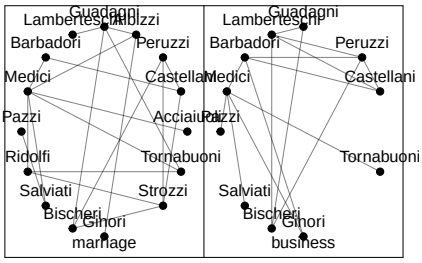
Wyniki działania programu przedstawiono na rysunku 2.

2.13 Przykładowe sieci

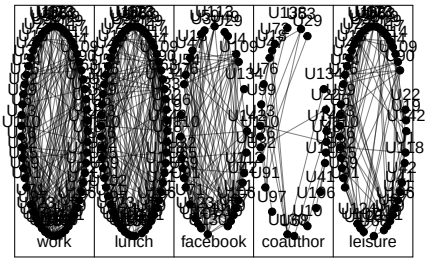
```
> ml_empty(name="unnamed")
Multilayer Network [0 actors, 0 layers, 0 vertices, 0 edges (0,0)]
> ml_aucs()
Multilayer Network [61 actors, 5 layers, 224 vertices, 620 edges (620,0)]
> ml_bankwiring()
Multilayer Network [14 actors, 6 layers, 62 vertices, 110 edges (110,0)]
> ml_florentine()
Multilayer Network [16 actors, 2 layers, 26 vertices, 35 edges (35,0)]
> ml_monastery()
Multilayer Network [18 actors, 10 layers, 175 vertices, 510 edges (510,0)]
> ml_tailorshop()
Multilayer Network [39 actors, 4 layers, 150 vertices, 552 edges (552,0)]
> ml_toy()
Multilayer Network [8 actors, 4 layers, 27 vertices, 30 edges (30,0)]
>
```



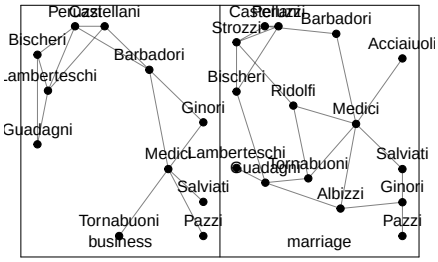
(a) AUCS network



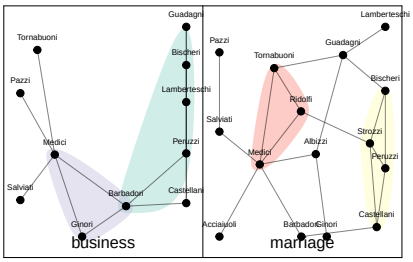
(b) Circular



(c) Circular 2



(d) Multiforce



(e) Clique percolation

Rysunek 2: Wizualizacje sieci

3 Badanie sieci wielowarstwowych zbudowane z sieci SmallWorld

Celem zadania było utworzenie dwóch sieci dwuwarstwowych (po 20 węzłów na każdej warstwie) na podstawie modelu teoretycznego WS+WS [1]. Jedna z tych sieci ma takie same rewiring probabilities na obu warstwatch, druga sieć różni się tym parametrem pomiędzy warstwami.

3.1 Utworzenie sieci

Rozpoczęto od przygotowania warstw sieci.

```
library(multinet)
library(igraph)

# Siec 1 takie same rewiring probabilities
ws1_1 <- sample_smallworld(dim=1, size=20, nei=1, p=0.1)
ws1_1 <- set_vertex_attr(ws1_1, "name", value = LETTERS[1:20])
ws1_2 <- sample_smallworld(dim=1, size=20, nei=1, p=0.1)
ws1_2 <- set_vertex_attr(ws1_2, "name", value = LETTERS[1:20])

# Siec 2 rne rewiring probabilities
ws2_1 <- sample_smallworld(dim=1, size=20, nei=1, p=0.1)
ws2_1 <- set_vertex_attr(ws2_1, "name", value = LETTERS[1:20])
ws2_2 <- sample_smallworld(dim=1, size=20, nei=1, p=0.8)
ws2_2 <- set_vertex_attr(ws2_2, "name", value = LETTERS[1:20])
```

Stworzono funkcję do tworzenia sieci z dwóch podanych warstw.

```
createML <- function(ws1, ws2) {
  net <- ml_empty()
  add_igraph_layer_ml(net, ws1, 'l1')
  add_igraph_layer_ml(net, ws1, 'l2')
  return(net)
}
```

Zdefiniowano funkcję która wprowadza określoną liczbę losowych połączeń pomiędzy warstwami.

```
addInterLayersEdges <- function(net, num) {
  edges <- data.frame(
    c(LETTERS[sample.int(10, num, replace=TRUE)]),
    c(rep("l1", num)),
    c(LETTERS[sample.int(10, num, replace=TRUE)]),
    c(rep("l2", num)))
  edges
  add_edges_ml(net, edges)
}
```

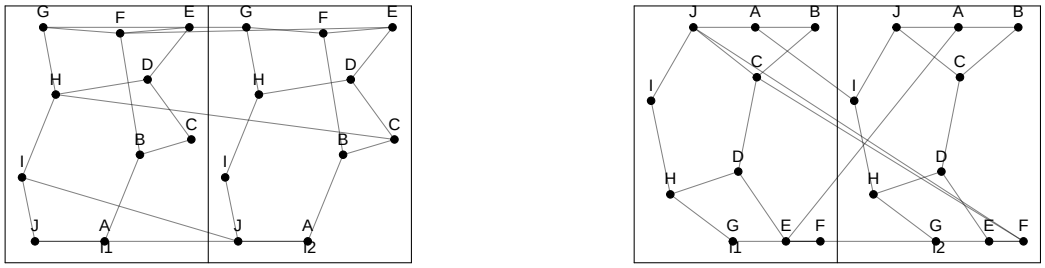
3.2 Badanie sieci z 5 losowymi połączeniami

3.2.1 Tworzenie sieci

Korzystamy z zdefiniowanych wcześniej funkcji.

```
net1 <- createML(ws1_1, ws1_2)
addInterLayersEdges(net1, 5)
plot(net1)

net2 <- createML(ws2_1, ws2_2)
addInterLayersEdges(net2, 5)
plot(net2)
```



(a) Sieć 1 (b) Sieć 2

Rysunek 3: Wygenerowane sieci – 5 losowych połaczeń

3.2.2 Wyznaczenie i porównanie podstawowych miar sieciowych w ujęciu wielowarstwowym

```
> plot(net2)
> degree_ml(net1)
 [1] 4 4 4 4 4 4 4 2 4 2 4 4 6 4 4 4 4 6 4 4
> degree_ml(net2)
 [1] 4 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4
```

3.2.3 Podobieństwo warstw

```
> layer_comparison_ml(net1,method="kulczynski2.actors")
   l1 l2
l1  1  1
l2  1  1
> layer_comparison_ml(net1,method="kulczynski2.edges")
   l1 l2
l1  1  1
l2  1  1
> layer_comparison_ml(net1,method="kulczynski2.triangles")
    l1  l2
l1 NaN NaN
l2 NaN NaN
>
> layer_comparison_ml(net1,method="dissimilarity.degree")
   l1 l2
l1  0  0
l2  0  0
> layer_comparison_ml(net1,method="KL.degree")
   l1 l2
l1  0  0
l2  0  0
> layer_comparison_ml(net1,method="jeffrey.degree")
   l1 l2
l1  0  0
l2  0  0
```

3.2.4 Porównanie społeczeności

```
> comm <- clique_percolation_ml(net1)
> length(comm)
```

```
[1] 3
> modularity_ml(net1, comm, gamma=1, omega=1)
[1] 0
>
> comm <- clique_percolation_ml(net2)
> length(comm)
[1] 3
> modularity_ml(net2, comm, gamma=1, omega=1)
[1] 0
```

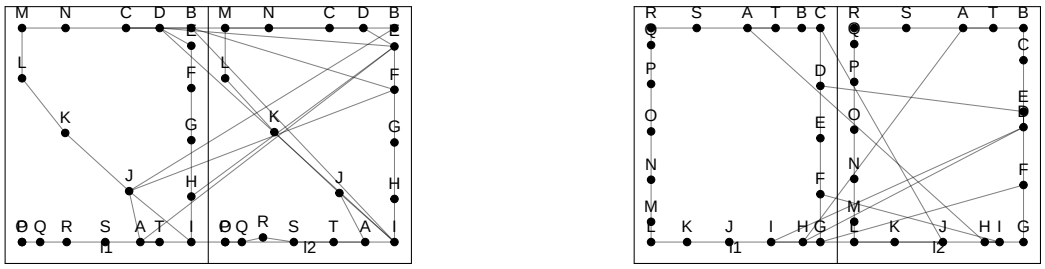
3.3 Badanie sieci z 10 losowymi połączeniami

3.3.1 Tworzenie sieci

Korzystamy z zdefiniowanych wcześniej funkcji.

```
net1 <- createML(ws1_1, ws1_2)
addInterLayersEdges(net1, 10)
plot(net1)

net2 <- createML(ws2_1, ws2_2)
addInterLayersEdges(net2, 10)
plot(net2)
```



(a) Sieć 1 (b) Sieć 2

Rysunek 4: Wygenerowane sieci – 10 losowych połaczeń

3.3.2 Wyznaczenie i porównanie podstawowych miar sieciowych w ujęciu wielowarstwowym

```
> degree_ml(net1)
 [1] 4 2 4 4 4 4 4 6 4 4 4 4 4 4 2 6 4 4 4 4
> degree_ml(net2)
 [1] 4 2 4 4 4 4 4 4 4 4 4 4 4 4 4 6 4 4 4 4
```

3.3.3 Podobieństwo warstw

```
> layer_comparison_ml(net1,method="kulczynski2.actors")
   l1 l2
l1  1  1
l2  1  1
> layer_comparison_ml(net1,method="kulczynski2.edges")
   l1 l2
```

```
l1  1  1
l2  1  1
> layer_comparison_ml(net1,method="kulczynski2.triangles")
    l1  l2
l1 NaN NaN
l2 NaN NaN
>
> layer_comparison_ml(net1,method="dissimilarity.degree")
   l1 l2
l1  0  0
l2  0  0
> layer_comparison_ml(net1,method="KL.degree")
   l1 l2
l1  0  0
l2  0  0
> layer_comparison_ml(net1,method="jeffrey.degree")
   l1 l2
l1  0  0
l2  0  0
```

3.3.4 Porównanie społeczności

```
> comm <- clique_percolation_ml(net1)
> length(comm)
[1] 3
> modularity_ml(net1, comm, gamma=1, omega=1)
[1] 0
>
> comm <- clique_percolation_ml(net2)
> length(comm)
[1] 3
> modularity_ml(net2, comm, gamma=1, omega=1)
[1] 0
```

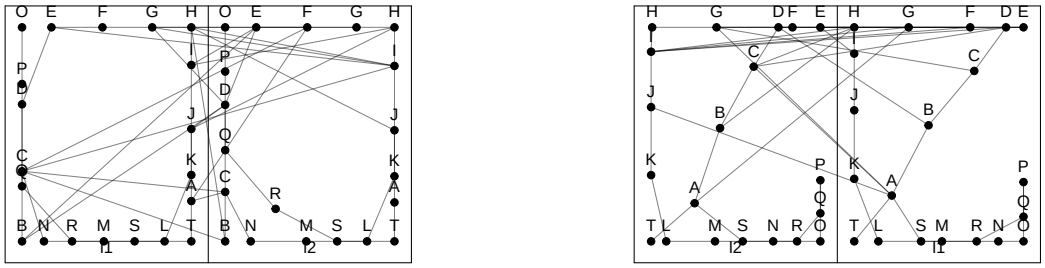
3.4 Badanie sieci z 20 losowymi połączeniami

3.4.1 Tworzenie sieci

Korzystamy z zdefiniowanych wcześniej funkcji.

```
net1 <- createML(ws1_1, ws1_2)
addInterLayersEdges(net1, 20)
plot(net1)

net2 <- createML(ws2_1, ws2_2)
addInterLayersEdges(net2, 10)
plot(net2)
```



(a) Sieć 1 (b) Sieć 2

Rysunek 5: Wygenerowane sieci – 20 losowych połaczeń

3.4.2 Wyznaczenie i porównanie podstawowych miar sieciowych w ujęciu wielowarstwowym

```
> degree_ml(net1)
 [1] 4 2 4 6 4 4 4 4 4 4 2 4 4 4 4 4 4 6 4 4 4 4
> degree_ml(net2)
 [1] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 6 2 4
```

3.4.3 Podobieństwo warstw

```
> layer_comparison_ml(net1,method="kulczynski2.actors")
   l1 l2
l1  1  1
l2  1  1
> layer_comparison_ml(net1,method="kulczynski2.edges")
   l1 l2
l1  1  1
l2  1  1
> layer_comparison_ml(net1,method="kulczynski2.triangles")
    l1  l2
l1 NaN NaN
l2 NaN NaN
>
> layer_comparison_ml(net1,method="dissimilarity.degree")
   l1 l2
l1  0  0
l2  0  0
> layer_comparison_ml(net1,method="KL.degree")
   l1 l2
l1  0  0
l2  0  0
> layer_comparison_ml(net1,method="jeffrey.degree")
   l1 l2
l1  0  0
l2  0  0
```

3.4.4 Porównanie społeczności

```
> comm <- clique_percolation_ml(net1)
> length(comm)
[1] 3
```

```
> modularity_ml(net1, comm, gamma=1, omega=1)
[1] 0
>
> comm <- clique_percolation_ml(net2)
> length(comm)
[1] 3
> modularity_ml(net2, comm, gamma=1, omega=1)
[1] 0
```

4 Podsumowanie

Zapoznano się z podstawowymi funkcjami dotyczącymi sieci wielowarstwowych. Przebadano sztucznie stworzoną sieć wielowarstwową dla różnych parametrów rewiring probabilities w modelu WS oraz dla różnej liczby połączeń międzywarstwowych.

Literatura

[1] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.