Assignment 1 - Network Modeling

Group 5

Lorenzo Benedetti (23-956-451) Francesco Freni (23-955-248)
Aristotelis Koutris (23-942-683) Ruben Oliveira Rodrigues (21-922-042)

2023-11-26

Task 1

(1)

- Out-degree:

$$s_{1i}(x) = \sum_j x_{ij}$$

- Reciprocity:

$$s_{2i}(x) = \sum_j x_{ij}x_{ji}$$

- Transitive reciprocated triplets effect:

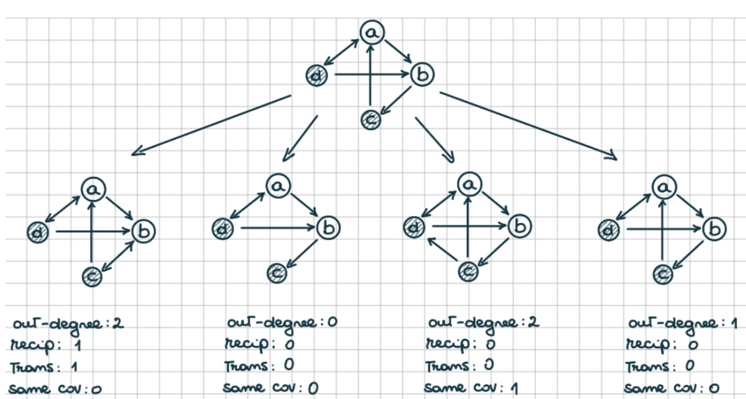
$$s_{3i}(x) = \sum_{j,h} x_{ij}x_{ji}x_{ih}x_{hj}$$

- Same covariate effect:

$$s_{4i}(x) = \sum_j x_{ij}I\{v_i = v_j\}$$

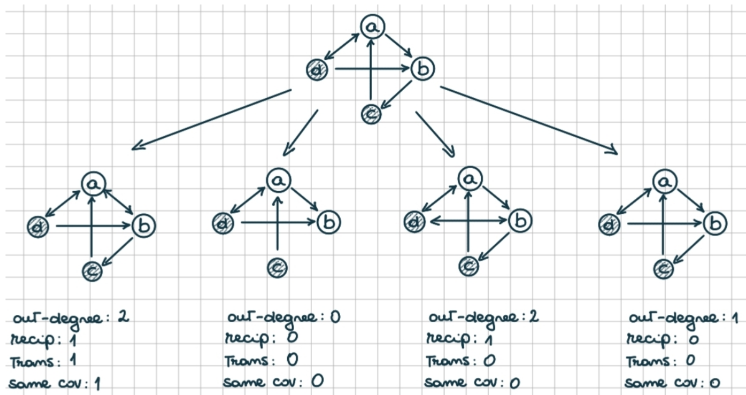
(2)

(i) Actor c can either add a tie to b, remove the tie to a, add a tie to d or keep the network unchanged.



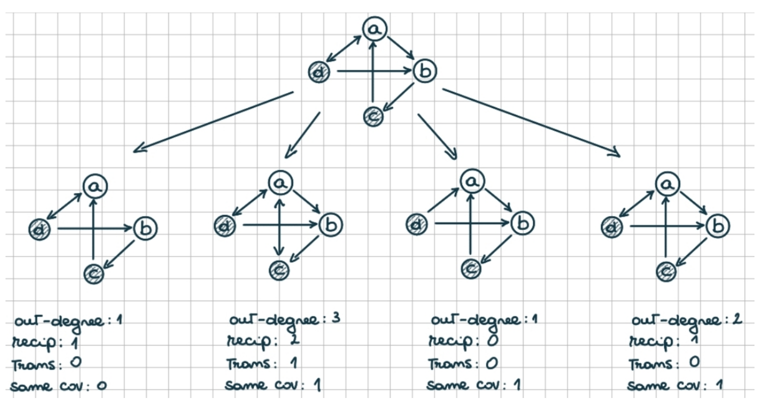
$$\frac{exp(-1.5\times2+2\times1+1\times1+1.5\times0)}{exp(-1.5\times2+2\times1+1\times1+1.5\times0)+exp(-1.5\times0+2\times0+1\times0+1.5\times0)+exp(-1.5\times2+2\times0+1\times0+1.5\times1)+exp(-1.5\times1+2\times0+1\times0+1.5\times0)} = 0.4087872$$

(ii) Actor b can either add a tie to a, remove the tie to c, add a tie to d or keep the network unchanged.



$$\frac{exp(-1.5\times2+2\times1+1\times1+1.5\times1)}{exp(-1.5\times2+2\times1+1\times1+1.5\times1)+exp(-1.5\times0+2\times0+1\times0+1.5\times0)+exp(-1.5\times2+2\times1+1\times0+1.5\times0)+exp(-1.5\times1+2\times0+1\times0+1.5\times0)} = 0.7380062$$

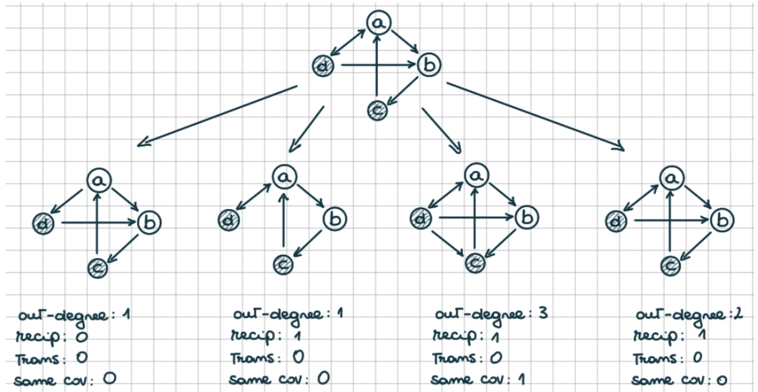
(iii) Actor a can either remove the tie to b, add a tie to c, remove the tie to d or keep the network unchanged.



$$\frac{exp(-1.5\times1+2\times1+1\times0+1.5\times0)}{exp(-1.5\times1+2\times1+1\times0+1.5\times0)+exp(-1.5\times3+2\times2+1\times1+1.5\times1)+exp(-1.5\times1+2\times0+1\times0+1.5\times1)+exp(-1.5\times2+2\times1+1\times0+1.5\times1)} = 0.1410791$$

(iv) Actor a can either remove the tie to a, remove the tie to b, add a tie to c or keep the network unchanged.

$$\frac{exp(-1.5\times2+2\times1+1\times0+1.5\times0)}{exp(-1.5\times1+2\times0+1\times0+1.5\times0)+exp(-1.5\times1+2\times1+1\times0+1.5\times0)+exp(-1.5\times3+2\times1+1\times0+1.5\times1)+exp(-1.5\times2+2\times1+1\times0+1.5\times0)} = 0.1410792$$



Task 2

(1)

```r
# Task 2.1 -------------------------------------------------------------------
# The function "simulation" simulates the network evolution between
# two time points.
# Given the network at time t1, denoted by x1, the function simulates the
# steps of the continuous-time Markov chain defined by a SAOM with outdegree,
# recip and transTrip statistics. Unconditional simulation is used.
# The function returns the network at time t2.
# The structure of the algorithm is described in the file
# _Simulating from SAOM.pdf_ available in
# the Lecture notes and additional material section on Moodle.

#' Simulate the network evolution between two time points
#'
#' @param n number of actors in the network
#' @param x1 network at time t1
#' @param lambda rate parameter
#' @param beta1 outdegree parameter
#' @param beta2 reciprocity parameter
#' @param beta3 transTrip parameter
#'
#' @return network at time t2
#'
#' @examples
#' netT1 <- matrix(c(
#'   0, 1, 0, 0, 0,
#'   0, 0, 0, 1, 0,
#'   0, 0, 0, 1, 1,
#'   1, 0, 1, 0, 0,
#'   0, 1, 1, 0, 1
#'   ),
#'   nrow = 5, ncol =  5, byrow = TRUE)
#' netT2 <- simulation(5, netT1, 4, -2, 0.5, 0.05)
simulation <- function(n, x1, lambda, beta1, beta2, beta3) {
  t <- 0 # time
  x <- x1
```

```r
  while (t < 1) {
    dt <- rexp(1, n * lambda)
    i <- sample(1:n, size=1)

    delta_outdegree <- rep(0, n)
    delta_rec <- rep(0, n)
    delta_trans_trip <- rep(0, n)
    for (j in (1:n)) {
      if (j==i) next
      delta_outdegree[j] <- 1 - 2*x[i,j]
      delta_rec[j] <- (1 - x[i,j])*x[j,i] - x[i,j]*x[j,i]
      #if ((x[i,j]==1) & (x[j,i]==1)) {
      #  delta_rec[j] <- -1
      #} else if (x[j,i] == 1) {
      #  delta_rec[j] <- 1
      #}
      for (h in 1:n) {
        delta_trans_trip[j] <- delta_trans_trip[j] + (1-x[i,j])*x[i,h]*x[h,j] -
          x[i,j]*x[i,h]*x[h,j]
        delta_trans_trip[j] <- delta_trans_trip[j] + x[i,h]*(1-x[i,j])*x[j,h] -
          x[i,h]*x[i,j]*x[j,h]
      }
    }
    p <- exp(beta1 * delta_outdegree + beta2 * delta_rec + beta3 * delta_trans_trip) /
      sum(exp(beta1 * delta_outdegree + beta2 * delta_rec + beta3 * delta_trans_trip))
    j <- which.max(rmultinom(1, 1, prob = p))
    if (i != j) x[i,j] <- 1 - x[i,j]
    t <- t + dt
  }
  return(x)
}
```

(2)

```r
net1 <- as.matrix(read.csv('net1.csv', header=F))
net2 <- as.matrix(read.csv('net2.csv', header=F))

waves <- sienaDependent(array(c(net1, net2), dim=c(22, 22, 2)))
myData <- sienaDataCreate(waves)
myData
```

```
## Dependent variables:  waves
## Number of observations: 2
##
## Nodeset                  Actors
## Number of nodes              22
##
## Dependent variable waves
## Type               oneMode
## Observations       2
## Nodeset            Actors
## Densities          0.17 0.17
```

```r
myeff <- getEffects(myData)
myeff <- includeEffects(myeff, transTrip)
```

```
##   effectName           include fix   test  initialValue parm
## 1 transitive triplets TRUE    FALSE FALSE          0    0
```

```r
myAlgorithm <- sienaAlgorithmCreate(
  nsub = 2, n3 = 3000, seed = 2023
)
```

```
## If you use this algorithm object, siena07 will create/use an output file Siena.txt .
```

```r
model0 <- siena07(myAlgorithm,
  data = myData, effects = myeff, returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4
)
model0
```

```
## Estimates, standard errors and convergence t-ratios
##
##                               Estimate   Standard   Convergence
##                                            Error      t-ratio
##
## Rate parameters:
##   0       Rate parameter       4.1444  ( 0.6933   )
##
## Other parameters:
##   1. eval outdegree (density) -1.1067  ( 0.1940   )   0.0158
##   2. eval reciprocity          0.4817  ( 0.3105   )   0.0379
##   3. eval transitive triplets  0.0774  ( 0.0932   )   0.0330
##
## Overall maximum convergence ratio:    0.0514
##
##
## Total of 3122 iteration steps.
```

```r
model0$rate
```

```
## [1] 4.144419
```

```r
model0$theta
```

```
## [1] -1.10667068  0.48170982  0.07740283
```