**Software Programs and Datasets for:**

**Deterministic blockmodeling of signed and two-mode networks: A tutorial with software**

**This document is divided into six sections. The first section describes the main Fortran programs and their executable (.exe) files, which are run at the command prompt. The second section describes corresponding Fortran subroutines and their dynamic-link-library (.dll) files, which are run via an R script within the R computing environment. The third section lists the network data files from the manuscript. The fourth section contains the content of the R scripts. The fifth section contains some computational experience with the methods for networks from the literature. The sixth section presents a signed network analysis for the Bitcoin alpha network.**

**I. Fortran Main Programs for Blockmodeling**

**A. Relocation Heuristic for Generalized Structural Balance**

**Fortran files**

rhgsbt2.for – Fortran source code

rhgsbt2.exe – executable file

**Input files**

datafil – an n × n signed network matrix

parset – a parameter file with three entries: number of objects, number of clusters, time limit

**Output files**

outpt – reports the criterion function value (inconsistencies) and the number of restarts

partition – the cluster assignments for the objects.

**Example**: To run the analysis for the ‘hafriend’ network, perform the following steps:

1) Place the Fortran, input, and hafriend files in a directory (e.g., Rblockmodeling).

2) Go to the command prompt and change the directory to Rblockmodeling:

> cd\Rblockmodeling

3) Edit the parset file for the number of objects, number of clusters, and time limit. For the hafriend network, we might use: 21 3 15, where 21 is the number of objects, 3 is the user-specified number of clusters, and 15 is the user-specified time limit in seconds.

4) Copy the file hafriend into datfil

> copy hafriend datafil

5) Run the program by typing rhgsbt2 at the command prompt:

> rhgsbt2

6) When the program finishes, two output files are created: outpt and partition. To see the criterion function value and the number of restarts for the relocation heuristic reported in the outpt file, type the following at the command prompt:

> type outpt

INCONSISTENCIES = 17.0000 RESTARTS = 279477

7) The file partition will have the cluster assignments for the 21 objects.

**B. Relocation Heuristic for Relaxed Structural Balance**

**Fortran files**

rhrsbt2.for – Fortran source code

rhrsbt2.exe – executable file

**Input files**

datafil – an n × n signed network matrix

parset – a parameter file with three entries: number of objects, number of clusters, time limit

**Output files**

outpt – reports the criterion function value (inconsistencies) and the number of restarts

partition – the cluster assignments for the objects.

**Example**: To run the analysis for the ‘sampsonT3.prn’ network, perform the following steps:

1) Place the Fortran, input, and sampsonT3.prn files in a directory (e.g., Rblockmodeling).

2) Go to the command prompt and change the directory to Rblockmodeling:

> cd\Rblockmodeling

3) Edit the parset file for the number of objects, number of clusters, and time limit. For the sampsonT3 network, we might use: 18 4 15, where 18 is the number of objects, 4 is the user-specified number of clusters, and 15 is the user-specified time limit in seconds.

4) Copy the file sampsonT3.prn into datfil

> copy sampsonT3.prn datafil

5) Run the program by typing rhrsbt2 at the command prompt:

> rhrsbt2

6) When the program finishes, two output files are created: outpt and partition. To see the criterion function value and the number of restarts for the relocation heuristic reported in the outpt file, type the following at the command prompt:

> type outpt

INCONSISTENCIES = 13.0000 RESTARTS = 177289

7) The file partition will have the cluster assignments for the 18 objects.

**C. Two-Mode Blockmodeling (Structural Equivalence) Heuristic**

**Fortran files**

tmklmed2.for – Fortran source code

tmklmed2.exe – executable file

**Input files**

datafil – an n × p binary network matrix

parset – a parameter file with five entries: number of row objects, number of column objects, number of row clusters, number of column clusters, time limit

**Output files**

outpt – reports the criterion function value (inconsistencies) and the number of restarts

rowpart – the cluster assignments for the row objects.

colpart – the cluster assignments for the column objects.

**Example**: To run the analysis for the ‘galask.prn’ network, perform the following steps:

1) Place the Fortran, input, and galask.prn files in a directory (e.g., Rblockmodeling).

2) Go to the command prompt and change the directory to Rblockmodeling:

> cd\Rblockmodeling

3) Edit the parset file for the number of row and column objects, number of row and column clusters, and time limit. For the galask network, we might use: 26 15 5 5 15, where 26 is the number of row objects, 15 is the number of column objects, 5 is the user-specified number of clusters, 5 is the user-specified number of column clusters, and 15 is the user-specified time limit in seconds.

4) Copy the file galask.prn into datfil

> copy galask.prn datafil

5) Run the program by typing tmklmed2 at the command prompt:

> tmklmed2

6) When the program finishes, two output files are created: outpt and partition. To see the criterion function value and the number of restarts for the heuristic reported in the outpt file, type the following at the command prompt:

> type outpt

INCONSISTENCIES = 47 RESTARTS = 112785

7) The file rowpart will have the cluster assignments for the 26 row objects (CEOs) and colpart will have the cluster assignments for the 15 column objects (clubs).

**D. Two-Mode *KL*-Means Blockmodeling Heuristic**

**Fortran files**

tmklm2.for – Fortran source code

tmklm2.exe – executable file

**Input files**

datafil – an n × p binary network matrix

parset – a parameter file with five entries: number of row objects, number of column objects, number of row clusters, number of column clusters, time limit

**Output files**

outpt – reports the criterion function value (VAF) and the number of restarts

rowpart – the cluster assignments for the row objects.

colpart – the cluster assignments for the column objects.

**Example**: To run the analysis for the ‘nyt.txt’ network, perform the following steps:

1) Place the Fortran, input, and nyt.txt files in a directory (e.g., Rblockmodeling).

2) Go to the command prompt and change the directory to Rblockmodeling:

> cd\Rblockmodeling

3) Edit the parset file for the number of row and column objects, number of row and column clusters, and time limit. For the nyt network, we might use: 108 25 9 5 60, where 108 is the number of row objects, 25 is the number of column objects, 9 is the user-specified number of clusters, 5 is the user-specified number of column clusters, and 60 is the user-specified time limit in seconds.

4) Copy the file nyt.txt into datfil

> copy nyt.txt datafil

5) Run the program by typing tmklm2 at the command prompt:

> tmklm2

6) When the program finishes, two output files are created: outpt and partition. To see the criterion function value and the number of restarts for the heuristic reported in the outpt file, type the following at the command prompt:

> type outpt

VAF = 0.54269893 RESTARTS = 35545

7) The file rowpart will have the cluster assignments for the 108 row objects and colpart will have the cluster assignments for the 25 column objects.

**E. A VNS Heuristic for Generalized Structural Balance**

For completeness, we provide the Fortran program used to implemented variable neighborhood search (VNS) for the Bitcoin Alpha network. The reference for this program is:

**Brusco, M. J., & Doreian, P. (2019). Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search. *Social Networks*, *56*, 70-80.** [**https://doi.org/10.1016/j.socnet.2018.08.007**](https://doi.org/10.1016/j.socnet.2018.08.007)

The relevant files are:

**Fortran files**

vnsgsbt2.for – Fortran source code

vnsgsbt2.exe – executable file

**Input files**

datafil – the signed network in ***sparse format***. The rows pertain to edges. The three entries in each row are the first vertex for the edge, the second vertex for the edge, and the edge weight.

parset – a parameter file with four entries: the number of vertices, the number of clusters, the number of edges, and the time limit (seconds).

**Output files**

outpt – reports the maximum number of ones in a row/column, the number of restarts for the relocation heuristic phase, the criterion function value (inconsistencies) for the relocation heuristic phase, and the criterion function value (inconsistencies) after VNS.

partition – the cluster assignments for the objects.

**Example**: To run the analysis for the ‘bitcoin.dat’ network, perform the following steps:

1) Place the Fortran, input, and bitcoin.dat files in a directory (e.g., Rblockmodeling).

2) Go to the command prompt and change the directory to Rblockmodeling:

> cd\Rblockmodeling

3) Edit the parset file for desired number of clusters and time limit. For the Bitcoin Alpha network, we might use: 3783 3 24186 300, where 3783 is the number of edges, 3 is the user-specified number of clusters, 24186 is the number of edges in the network, and 300 is the user-specified time limit in seconds.

4) Copy the file Bitcoin.dat into datfil

> copy bintcoin.dat datafil

5) Run the program by typing vnsgsbt2 at the command prompt:

> vnsgsbt2

6) When the program finishes, two output files are created: outpt and partition. To see the criterion function values and the number of restarts for the relocation heuristic reported in the outpt file, type the following at the command prompt:

> type outpt

MAXIMUM NONZEROS IN ROWS OR COLUMNS 491

RELOCATION HEURISTIC TOTAL RESTARTS 17116

RELOCATION HEURISTIC INCONSITENCIES 4287.0

VARIABLE NEIG SEARCH INCONSITENCIES 4239.0

7) The file labeled partition will have the cluster assignments for the 3783 vertices.

**II. Fortran Subroutines for Blockmodeling in R**

**A. Relocation Heuristic for Generalized Structural Balance**

rhgsbt.for – Fortran source code

rhgsbt.dll – dynamic-link-library executable file

rhgsbt.R – R script to run rhgsbt.dll (note: the call to access rhgsbt.dll in the script obtains the dll from the Rblockmodeling directory where it is located on my computer).

**Inputs**: An N × N signed network matrix (A), a desired number of clusters (C : 1 < C < N), and a time limit in seconds (TLIMIT).

**Outputs**: The objective function value (obj), the partition (P), and the number of restarts (restarts).

**Example**: The following commands load and define the ‘hafriend’ network, set the desired, number of clusters and time limit, run rhgsbt, and display the output.

> A <- read.table("c:/Rblockmodeling/hafriend")

> A <- as.matrix(A)

> C <- 3

> TLIMIT <- 15

> source("C:\\Users\\michael brusco\\Documents\\rhgsbt.R")

> rhgsbt(A,C,TLIMIT)

> obj

[[1]]

[1] 17

> restarts

[[1]]

[1] 428609

> P

[,1]

[1,] 3

[2,] 3

[3,] 2

[4,] 3

[5,] 3

[6,] 3

[7,] 1

[8,] 1

[9,] 2

[10,] 3

[11,] 3

[12,] 2

[13,] 3

[14,] 2

[15,] 1

[16,] 3

[17,] 3

[18,] 2

[19,] 3

[20,] 2

[21,] 1

>

**B. Relocation Heuristic for Relaxed Structural Balance**

rhrsbt.for – Fortran source code

rhrsbt.dll – dynamic-link-library executable file

rhrsbt.R – R script to run rhrsbt.dll (note: the call to access rhrsbt.dll in the script obtains the dll from the Rblockmodeling directory where it is located on my computer).

**Inputs**: An N× N signed network matrix (A), a desired number of clusters (C : 1 < C < N), and a time limit in seconds (TLIMIT).

**Outputs**: The objective function value (obj), the partition (P), and the number of restarts (restarts).

**Example**: The following commands load and define the ‘sampsonT3’ network, set the desired, number of clusters and time limit, run rhrsbt, and display the output.

> A <- read.table("c:/Rblockmodeling/sampsonT3.prn")

> A <- as.matrix(A)

> C <- 4

> TLIMIT <- 15

> source("C:\\Users\\michael brusco\\Documents\\rhrsbt.R")

> A < as.matrix(A)

> rhrsbt(A,C,TLIMIT)

> obj

[[1]]

[1] 13

> restarts

[[1]]

[1] 228822

> P

[,1]

[1,] 2

[2,] 2

[3,] 4

[4,] 3

[5,] 1

[6,] 3

[7,] 2

[8,] 3

[9,] 1

[10,] 3

[11,] 3

[12,] 2

[13,] 4

[14,] 2

[15,] 2

[16,] 2

[17,] 4

[18,] 4

>

**C. Two-Mode Blockmodeling (Structural Equivalence) Heuristic**

tmklmed.for – Fortran source code

tmklmed.dll – dynamic-link-library executable file

tmklmed.R – R script to run tmklmed.dll (note: the call to access tmklmed.dll in the script obtains the dll from the Rblockmodeling directory where it is located on my computer).

**Inputs**: An *RO* × *CO* two-mode binary network matrix (A), a desired number of clusters for row objects (RC : 1 < RC < RO), a desired number of clusters for row objects (CC : 1 < CC < CO), and a time limit in seconds (TLIMIT).

**Outputs**: The objective function value (objval), the partition of row objects (RP), the partition of column objects (CP), and the number of restarts (restarts).

**Example**: The following commands load and define the ‘galask’ network, set the desired, number of clusters for row and column clusters, set the time limit, run tmklmed, and display the output.

> A <- read.table("c:/Rblockmodeling/galask.prn")

> A <- as.matrix(A)

> RC <- 5

> CC <- 5

> TLIMIT <- 30

> source("C:\\Users\\michael brusco\\Documents\\tmklmed.R")

> tmklmed(A,RC,CC,TLIMIT)

> objval

[[1]]

[1] 47

> restarts

[[1]]

[1] 344223

> RP

[,1]

[1,] 5

[2,] 5

[3,] 1

[4,] 1

[5,] 3

[6,] 4

[7,] 5

[8,] 4

[9,] 4

[10,] 3

[11,] 4

[12,] 1

[13,] 4

[14,] 2

[15,] 1

[16,] 2

[17,] 1

[18,] 3

[19,] 1

[20,] 4

[21,] 2

[22,] 1

[23,] 1

[24,] 2

[25,] 2

[26,] 3

> CP

[,1]

[1,] 3

[2,] 4

[3,] 5

[4,] 1

[5,] 2

[6,] 1

[7,] 2

[8,] 2

[9,] 2

[10,] 2

[11,] 1

[12,] 2

[13,] 2

[14,] 2

[15,] 2

>

**D. Two-Mode KL-Means Heuristic**

tmklm.for – Fortran source code

tmklm.dll – dynamic-link-library executable file

tmklm.R – R script to run tmklm.dll (note: the call to access tmklm.dll in the script obtains the dll from the Rblockmodeling directory where it is located on my computer).

**Inputs**: An *RO* × *CO* two-mode network matrix (A), a desired number of clusters for row objects (RC : 1 < RC < RO), a desired number of clusters for row objects (CC : 1 < CC < CO), and a time limit in seconds (TLIMIT).

**Outputs**: The variance-accounted-for (vaf), the partition of row objects (RP), the partition of column objects (CP), and the number of restarts (restarts).

**Example**: The following commands load and define the ‘nyt.txt’ network, set the desired, number of clusters for row and column clusters, set the time limit, run tmklm, and display the output (I suppressed the partition of the row objects because there are 108 objects).

> A <- read.table("c:/Rblockmodeling/nyt.txt")

> A <- as.matrix(A)

> RC <- 9

> CC <- 5

> TLIMIT <- 60

> source("C:\\Users\\michael brusco\\Documents\\tmklm.R")

> tmklm(A,RC,CC,TLIMIT)

> vaf

[[1]]

[1] 0.5426989

> restarts

[[1]]

[1] 33767

> CP

[,1]

[1,] 5

[2,] 5

[3,] 5

[4,] 5

[5,] 4

[6,] 4

[7,] 4

[8,] 4

[9,] 4

[10,] 1

[11,] 1

[12,] 1

[13,] 1

[14,] 1

[15,] 1

[16,] 2

[17,] 2

[18,] 2

[19,] 2

[20,] 2

[21,] 3

[22,] 3

[23,] 3

[24,] 3

[25,] 3

>

**III. *Datasets***

haddate – House A, double-dating relation (Lemann & Solomon, 1952)

hafriend – House A, friendship relation (Lemann & Solomon, 1952)

haweekend – House A, weekend relation (Lemann & Solomon, 1952)

haroommate – House A, roommate (Lemann & Solomon, 1952)

housea.prn – House A, aggregated relations (Lemann & Solomon, 1952)

hbddate – House B, double-dating relation (Lemann & Solomon, 1952)

hbfriend – House B, friendship relation (Lemann & Solomon, 1952)

hbweekend – House B, weekend relation (Lemann & Solomon, 1952)

hbroommate – House B, roommate (Lemann & Solomon, 1952)

houseb.prn – House B, aggregated relations (Lemann & Solomon, 1952)

hcddate – House C, double-dating relation (Lemann & Solomon, 1952)

hcfriend – House C, friendship relation (Lemann & Solomon, 1952)

hcweekend – House C, weekend relation (Lemann & Solomon, 1952)

hcroommate – House C, roommate (Lemann & Solomon, 1952)

housec.prn – House C, aggregated relations (Lemann & Solomon, 1952)

sampsonT2.prn – Sampson (1968) monastery network, time 2

sampsonT3.prn – Sampson (1968) monastery network, time 3

sampsonT4.prn – Sampson (1968) monastery network, time 4

sampsonAgg – Sampson (1968) monastery network, aggregated

bwr.prn – backboard wiring room network (Roethlisberger & Dickson, 1939)

bales.prn – Bales (1970) network

newcomb – Newcomb’s (1961) fraternity network, week 15

mckinney.txt – Mckinney (1948) network

galask.prn – Galaskiewicz’ (1985) CEO/Club network

nyt.txt – Turning Point Project network (Brusco & Doreian, 2015)

depanx.prn – depression/anxiety network (Forbes et al., 2017)

bitcoin.dat - BitcoinAlpha network (Kumar et al., 2016) in sparse form

bitcoinalpha – BitcoinAlpha network as 3783 × 3783 binary matrix

**IV. *R* scripts for blockmodeling**

The R scripts for each of the four blockmodeling algorithms used in this paper are provided below. For each script, the primary engine is a Fortran .dll file of the same name that does of all the computational lifting.

**rhgsbt.R – Relocation Heuristic for Generalized Structural Balance**

rhgsbt = function(A,C,TLIMIT) {

# This program runs a relocation heuristic for generalized structural balance

# partitioning of an N x N signed matrix

# INPUTS

# A - an N x N signed network matrix

# C - the number of cluster (1 < C < N)

# TLIMIT - a desired time limit

# OUTPUTS

# obj - the Doreian & Mrvar objective value

# P - and N-dimensional vector of cluster assignments

# restarts - the number of restarts within the time limit

N = dim(A)[1]

OBJVAL = 0

NREPS = 0

EB <- matrix(0, nrow = N, ncol = 1)

dyn.load("c:/RBlockmodeling/rhgsbt.dll")

res=.Fortran("rhgsbt",as.integer(N),as.integer(C),as.double(TLIMIT),as.double(OBJVAL),as.integer(A),as.integer(EB),as.integer(NREPS))

p <- as.numeric(as.character(unlist(res[6])))

P <<- matrix(p, nrow = N, ncol = 1)

obj <<- res[4]

restarts <<- res[7]

}

**rhrsbt.R – Relocation Heuristic for Relaxed Structural Balance**

rhrsbt = function(A,C,TLIMIT) {

# This program runs a relocation heuristic for generalized structural balance

# partitioning of an N x N signed matrix

# INPUTS

# A - an N x N signed network matrix

# C - the number of cluster (1 < C < N)

# TLIMIT - a desired time limit

# OUTPUTS

# obj - the Doreian & Mrvar objective value

# P - and N-dimensional vector of cluster assignments

# restarts - the number of restarts within the time limit

N = dim(A)[1]

OBJVAL = 0

NREPS = 0

EBEST <- matrix(0, nrow = N, ncol = 1)

dyn.load("c:/RBlockmodeling/rhrsbt.dll")

res=.Fortran("rhrsbt",as.integer(N),as.integer(C),as.double(TLIMIT),as.double(OBJVAL),as.integer(A),as.integer(EBEST),as.integer(NREPS))

p <- as.numeric(as.character(unlist(res[6])))

P <<- matrix(p, nrow = N, ncol = 1)

obj <<- res[4]

restarts <<- res[7]

}

**tmklmed.R – Relocation Heuristic for Two-Mode Blockmodeling (Structural Equivalence)**

tmklmed = function(A,RC,CC,TLIMIT) {

# This program runs two-mode KL-medians for an

# an RO x CO binary network matrix.

# INPUTS

# A - an RO x CO two-mode binary network matrix

# RC - the number of row clusters (1 < RC < RO)

# CC - the number of row clusters (1 < CC < CO)

# TLIMIT - a desired time limit

# OUTPUTS

# objval - total number of inconsistencies

# RP - an RO-dimensional vector of row cluster assignments

# CP - an CO-dimensional vector of column cluster assignments

# restarts - the number of restarts within the time limit

RO = dim(A)[1]

CO = dim(A)[2]

GBEST = 0

NREPS = 0

GR <- matrix(0, nrow = RO, ncol = 1)

GC <- matrix(0, nrow = CO, ncol = 1)

dyn.load("c:/RBlockmodeling/tmklmed.dll")

res =.Fortran("tmklmed",as.integer(RO),as.integer(CO),as.integer(RC),as.integer(CC),as.double(TLIMIT),as.integer(A),as.integer(GR),as.integer(GC),as.integer(GBEST),as.integer(NREPS))

p <- as.numeric(as.character(unlist(res[7])))

RP <<- matrix(p, nrow = RO, ncol = 1)

q <- as.numeric(as.character(unlist(res[8])))

CP <<- matrix(q, nrow = CO, ncol = 1)

objval <<- res[9]

restarts <<- res[10]

}

**tmklm.R – Relocation Heuristic for Two-Mode KL-means**

tmklm = function(A,RC,CC,TLIMIT) {

# This program runs two-mode K-means for an

# an RO x CO network matrix.

# INPUTS

# A - an RO x CO two-mode network matrix

# RC - the number of row clusters (1 < RC < RO)

# CC - the number of row clusters (1 < CC < CO)

# TLIMIT - a desired time limit

# OUTPUTS

# vaf - the variance accounted

# RP - an RO-dimensional vector of row cluster assignments

# CP - an CO-dimensional vector of column cluster assignments

# restarts - the number of restarts within the time limit

RO = dim(A)[1]

CO = dim(A)[2]

VAF = 0

NREPS = 0

RBEST <- matrix(0, nrow = RO, ncol = 1)

CBEST <- matrix(0, nrow = CO, ncol = 1)

dyn.load("c:/RBlockmodeling/tmklm.dll")

res=.Fortran("tmklm",as.integer(RO),as.integer(CO),as.integer(RC),as.integer(CC),as.double(TLIMIT),as.double(A),as.integer(RBEST),as.integer(CBEST),as.double(VAF),as.integer(NREPS))

p <- as.numeric(as.character(unlist(res[7])))

RP <<- matrix(p, nrow = RO, ncol = 1)

q <- as.numeric(as.character(unlist(res[8])))

CP <<- matrix(q, nrow = CO, ncol = 1)

vaf <<- res[9]

restarts <<- res[10]

}

**V. *Computational results for some empirical networks***

To assure that the blockmodeling programs produced acceptable results, we conducted experimental tests using a variety of empirical networks drawn from the literature. The algorithms were implemented on a desktop computer with an Intel® Core™ i7-6700T CPU at 2.8 GHz with 16GB of RAM. The results of these experimental analyses are distinguished on the basis of whether the empirical data are one-mode signed matrices or two-mode matrices.

***A. Results for signed networks***

The 23 signed networks used in our experimental analyses were taken from six different studies. These data have been analyzed by numerous authors over several decades. In many cases, benchmark optimal solutions are available from previous studies describing exact solution methods for generalized and/or relaxed structural balances (Brusco et al., 2011; Brusco & Steinley, 2010; Figuieredo & Moura, 2013). The networks are as follows:

(1) Bales group dynamics network (Bales, 1970) - a signed network (Bales) obtained from a group dynamics study that features positive and negative affect ties among *n* = 20 co-workers.

(2) Bank wiring room network (Roethlisberger & Dickson, 1939) – a signed network (BWR) obtained from positive and negative relations among *n* = 14 bank wiring room workers.

(3) House networks (Lemann & Solomon, 1952) – 15 signed network matrices (17 ≤ *n* ≤ 21) obtained from three different sorority houses (A, B, and C) at an eastern college. For each house, measurements were taken for four signed preference relations regarding; (1) double-dating behavior - *dd*, (2) maintaining friendship after college - *fr*, (3) establishing a roommate relationship – *rm*, and (4) setting up a weekend visit with family - *we*. A fifth network for each house was obtained by summing across the four measured relations (Doreian, 2008).

(4) Pseudo-fraternity network (Doreian, Kapuscinski, Krackhardt, & Szczypula, 1996; Newcomb, 1961; Nordlie, 1958) – a signed network among *n* = 17 male college students based on affect rankings.

(5) Monastery networks (Sampson, 1968) – four signed networks pertaining to affect ties among *n* = 18 monks. Three of the networks (Sampson T2, Sampson T2, and Sampson T4) correspond to affect measurements taken at three different points in time. The fourth (Sampson Ag) is an aggregation of these three matrices obtained via summation (see Doreian & Mrvar, 1996).

(6) Schoolchildren network (McKinney, 1948) – a signed network associated with collaboration “willingness or unwillingness” ties among *n* = 29 schoolchildren.

For each of the 23 signed networks, the rhgsbt and rhrsbt algorithms described above were implemented for the range of 2 ≤ *K* ≤ 8. For each network, each algorithm, and each *K*, a time limit of 15 seconds was used. The results are reported in Table 1. It is evident from the table that the value of *Z*1(π) typically reaches a minimum on the interval 3 ≤ *K* ≤ 5, consistent with Theorem 10.6 (of Doreian et al. (2005) regarding the behavior of *Z*1(π); however, *Z*2(π) is monotonically nonincreasing as *K* increases consistent with Theorem 4 of Doreian and Mrvar (2009).

Table 1. Results for rhgsbt and rhrsbt (15 second time limit) for empirical signed networks.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | rhgsbt – *Z*1(π) for 2 ≤ *K* ≤ 8 | | | | | | |  | rhrsbt – *Z*2(π) for 2 ≤ *K* ≤ 8 | | | | | | |
|  | *N* | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Bales | 20 | 32 | 28 | **26** | 28 | 30 | 32 | 34 |  | 28 | 18 | 14 | 10 | 6 | 4 | 2 |
| BWR | 14 | 4 | 2 | **0** | 0 | 0 | 2 | 4 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| House A - dd | 21 | 25 | 18 | **17** | 17 | 18 | 19 | 21 |  | 25 | 15 | 10 | 6 | 4 | 3 | 1 |
| House A -rm | 21 | 20 | 15 | **13** | 13 | 14 | 16 | 19 |  | 18 | 12 | 6 | 4 | 2 | 1 | 0 |
| House A -we | 21 | 23 | 18 | 15 | **14** | 14 | 16 | 19 |  | 20 | 8 | 4 | 3 | 1 | 0 | 0 |
| House A -fr | 21 | 25 | 17 | **15** | 15 | 16 | 17 | 19 |  | 21 | 9 | 4 | 3 | 2 | 1 | 0 |
| House A -sum | 21 | 97 | 74 | **64** | 65 | 66 | 74 | 82 |  | 96 | 50 | 31 | 27 | 21 | 17 | 12 |
| House B -dd | 17 | 24 | 20 | **18** | 19 | 20 | 22 | 24 |  | 19 | 16 | 11 | 8 | 5 | 3 | 2 |
| House B -rm | 17 | 25 | 18 | **17** | 19 | 20 | 22 | 24 |  | 18 | 12 | 10 | 6 | 5 | 2 | 1 |
| House B -we | 17 | 23 | **19** | 19 | 20 | 21 | 23 | 25 |  | 21 | 17 | 12 | 8 | 6 | 3 | 1 |
| House B -fr | 17 | 21 | 19 | **18** | 19 | 20 | 21 | 23 |  | 21 | 14 | 10 | 7 | 4 | 3 | 2 |
| House B -sum | 17 | 103 | 84 | **81** | 83 | 87 | 92 | 100 |  | 84 | 69 | 56 | 43 | 33 | 28 | 22 |
| House C -dd | 20 | 16 | **13** | 13 | 14 | 16 | 19 | 21 |  | 16 | 12 | 9 | 7 | 5 | 3 | 1 |
| House C -rm | 20 | 23 | **18** | 18 | 18 | 20 | 23 | 25 |  | 23 | 14 | 12 | 9 | 6 | 4 | 2 |
| House C -we | 20 | 15 | **14** | 15 | 16 | 18 | 20 | 22 |  | 15 | 11 | 8 | 6 | 4 | 2 | 1 |
| House C -fr | 20 | 16 | **14** | 14 | 16 | 18 | 20 | 23 |  | 16 | 13 | 9 | 6 | 5 | 3 | 1 |
| House C -sum | 20 | 64 | **59** | 59 | 63 | 69 | 79 | 89 |  | 64 | 53 | 43 | 35 | 29 | 24 | 19 |
| Newcomb | 17 | 27 | 22 | **20** | 21 | 25 | 29 | 33 |  | 10 | 7 | 5 | 3 | 1 | 0 | 0 |
| Sampson T2 | 18 | 43 | **35** | 38 | 41 | 45 | 50 | 54 |  | 43 | 25 | 13 | 8 | 4 | 2 | 1 |
| Sampson T3 | 18 | 32 | **22** | 27 | 32 | 37 | 42 | 47 |  | 32 | 21 | 13 | 8 | 6 | 4 | 2 |
| Sampson T4 | 18 | 25 | **21** | 25 | 30 | 35 | 41 | 48 |  | 25 | 21 | 10 | 6 | 4 | 1 | 0 |
| Sampson Ag | 18 | 86 | **62** | 71 | 80 | 99 | 113 | 135 |  | 86 | 54 | 36 | 25 | 16 | 12 | 8 |
| McKinney | 29 | **12** | 12 | 14 | 18 | 28 | 38 | 48 |  | 8 | 2 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Note: The bolded elements for rhgsbt are the first instances for which the criterion function reached a minimum value.

The rhgsbt results for the BWR, Bales, and Sampson T2, Sampson T3, and Sampson T4 networks match the known globally-optimal value for *Z*1(π) reported by Brusco and Steinley (2010) for all 2 ≤ *K* ≤ 8. For the other 18 networks, some benchmarking is possible using the results reported by Figueiredo and Moura (2013, p. 644). Although these authors do not report optimal values of *Z*1(π) for different values of *K*; they do report the optimal objective function value for the correlation clustering problem, which corresponds to the minimum value of *Z*1(π) across all *K*. In all instances, the minimum value of *Z*1(π) across all *K* obtained by rhgsbt matched the globally-optimal objective function value for the correlation clustering problem. In some instances (House B we, House C fr, House C rm), our results show that the optimal objective function value for the correlation clustering problem can be achieved in fewer clusters than reported by Figueiredo and Moura (2013, p. 644).

The objective function used by Brusco et al. (2011) for relaxed structural balance is equal to one-half of the value of *Z*2(π). Using this relationship, the rhrsbt results for the House A – sum, House B – sum, House C – sum, Sampson T4, Newcomb, and Mckinney networks match the known globally-optimal value for *Z*2(π) reported by Brusco et al. (2011) for all 2 ≤ *K* ≤ 7 (Brusco et al. 2011 did not consider *K* = 8). For some of the other networks, we again draw from the results reported by Figueiredo and Moura (2013). For example, the rhrsbt results for Sampson T2, Sampson T3, and Sampson T4 are as always as good as those reported by Figueiredo and Moura (2013, p. 646). In several instances (Sampson T2 at *K* = 4, Sampson T3 at *K* = 6, and Sampson T3 at *K* = 7), the objective function value obtained by rhrsbt is better than the one reported by Figueiredo and Moura (2013).

***B. Results for two-mode networks***

Relative to signed one-mode networks, there are far fewer reported globally-optimal solutions for two-mode networks. One exception is the provision of optimal solutions for the CEO/Club affiliation network (Galaskiewicz, 1985) by Brusco et al. (2013). This network pertains to the affiliation (*bij* = 1) or lack of affiliation (*bij* = 0) of *n* = 26 CEOs in *p* = 15 clubs. We ran the tmklmed algorithm for the CEO Club network for all combinations of *K* and *L* on the intervals 2 ≤ *K* ≤ 5 and 2 ≤ *L* ≤ 6, using a 15-second time limit for each combination. The values of *Z*3(π, ω) for each combination are reported in the top panel of Table 2. For each combination, the value of *Z*3(π, ω) obtained by the tmkmed algorithm matched the known globally-optimal value published by Brusco et al. (2013).

Table 2. The top panel contains tmklmed *Z*3(π,ω) results for the CEO/club network for a 15-second time limit. The bottom panel contains tmklm *Z*4(π,ω) results for the Turning Point Project network for a 60-second time limit.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *n* | *p* |  | *L* = 2 | *L =* 3 | *K* = 4 | *L* = 5 | *L* = 6 |
| CEO/club | 26 | 15 | *K* = 2 | 73 | 66 | 62 | 62 | 62 |
| tmklmed | 26 | 15 | *K* = 3 | 73 | 64 | 58 | 56 | 56 |
| (15 seconds) | 26 | 15 | *K* = 4 | 73 | 61 | 54 | 51 | 51 |
|  | 26 | 15 | *K* = 5 | 73 | 61 | 52 | 47 | 46 |
|  |  |  |  |  | *L =* 3 | *K* = 4 | *L* = 5 | *L* = 6 |
| TPP | 108 | 25 | *K* = 3 |  | .2728686 | .2831353 | .2893438 | .2928794 |
| tmklm | 108 | 25 | *K* = 4 |  | .3256796 | .3442926 | .3560634 | .3589723 |
| (60 seconds) | 108 | 25 | *K* = 5 |  | .3661490 | .3972031 | .4083790 | .4129656 |
|  | 108 | 25 | *K* = 6 |  | .3875883 | .4395894 | .4556974 | .4604928 |
|  | 108 | 25 | *K* = 7 |  | .3996407 | .4612368 | .5017758 | .5074852 |
|  | 108 | 25 | *K* = 8 |  | .4109332 | .4762076 | .5263150 | .5335471 |
|  | 108 | 25 | *K* = 9 |  | .4197877 | .4876569 | .5426989 | .5514885 |
|  | 108 | 25 | *K* = 10 |  | .4265217 | .4970661 | .5563876 | .5674802 |

Note: The top panel reports values of the number of inconsistencies criterion function (*Z*3(π, ω)) and

the lower panel values for variance-accounted-for (*Z*4(π, ω))

In light of the dearth of published globally-optimal solutions for two-mode networks using the two-model *KL*-means criterion, we selected a network for which the results obtained by a metaheuristic have been published. The network data pertain to the Turning Point Project (TPP), which involved the signing of *p* = 25 environmentally-oriented advertisements in the New York Times (NYT) during 1999-2000 by *n* = 108 organizations. Each organization signed one or more of the advertisements and the elements of **B** are *bij* = 1 if organization *i* signed advertisement *j* and *bij* = 0 otherwise. Brusco and Doreian (2015) obtained two-model *KL*-means solutions for this network using both a relocation heuristic and a genetic algorithm for all combinations of *K* and *L* on the intervals 3 ≤ *K* ≤ 10 and 3 ≤ *L* ≤ 6. We ran the tmklm algorithm for the TPP network for each of these combinations using a 60-second time limit for each combination. The results are reported in the bottom panel of Table 2.

For 18 of the 32 combinations of *K* and *L*, the value of *VAF*(π, ω) obtained by the tmklm algorithm matched the value obtained by the two-mode *KL*-means heuristic implemented by Brusco and Doreian (2015). For the remaining 14 combinations, the tmklm algorithm provided a larger (better) value of *VAF*(π, ω). Somewhat surprisingly, the tmklm algorithm also performed well relative to the genetic algorithm implementation of Brusco and Doreian (2015), yielding a better *VAF*(π, ω) value for five combinations, a worse *VAF*(π, ω) value for two combinations, and the same *VAF*(π, ω) value for 25 combinations.

**VI. *Example: The Bitcoin Alpha (signed) trust network***

***A. The data***

This example pertains to structural balance partitioning of a *trust* network. In industrial and organizational psychology, trust networks commonly arise in the context of trust and/or leader-follower relationships within the workplace (Goodwin, Bowler, & Whittington, 2009; Lau & Liden, 2008). However, in our demonstration, we focus on a large, signed, and weighted trust network that corresponds to Bitcoin trading (Kumar, Spezzano, Subrahmanian, & Faloutsos, 2016). Bitcoin is a cryptocurrency that is commonly used among a group (or *exchange*) of traders on the Internet. Because the trading process associated with Bitcoin is anonymous, there is a potential risk undertaken by the trading parties and, accordingly, trust is an important aspect of a Bitcoin exchange.

Kumar et al. (2016) constructed a trust network among *n* = 3783 participants in a Bitcoin exchange (known as Bitcoin Alpha). Participants rated other members of the exchange on an integer scale ranging from -10 to 10 (excluding the use of a rating of 0), where -10 represents a high level of distrust (a perception that the target actor is fraudulent) and a 10 a high level of trust (a perception that the target actor is as trustworthy as oneself). The resulting network consists of 24,186 signed edges (93% of the edges are positive). Although the Bitcoin Alpha has been analyzed previously in the context of edge prediction, to the best of our knowledge it has not been analyzed using generalized or relaxed structural balance blockmodeling. Our goal here is to provide a such a blockmodeling analysis of the data.

***B. Model selection***

We implemented generalized structural balance partitioning in R for the Bitcoin Alpha network using the rhgsbt program. The model was implemented for values of *K* on the interval 2 ≤ *K* ≤ 6 using a time limit of 300 seconds for each number of clusters. The objective function value *Z*1(π) and total number of restarts were recorded. The results are displayed in Table 3.

Table 3. Criterion function values and number of restarts for the Bitcoin Alpha network using rhgsbt and variable neighborhood search.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | *K* = 2 | *K* = 3 | *K* = 4 | *K* = 5 | *K* = 6 |
|  |  |  |  |  |  |  |
| rhgsbt | *Z*1(π) | 4494 | 4305 | 4300 | 4297 | 4291 |
| 300 seconds | restarts | 351 | 202 | 136 | 101 | 84 |
|  |  |  |  |  |  |  |
| VNS | *Z*1(π) - RH | 4421 | 4287 | 4280 | 4279 | 4268 |
| 150 seconds relocation | restarts | 33944 | 18423 | 11977 | 8743 | 6814 |
| 150 seconds search | *Z*1(π) - VNS | 4366 | 4239 | 4247 | 4247 | 4248 |
|  |  |  |  |  |  |  |
| VNS | *Z*1(π) - RH | 4421 | 4280 | 4269 | 4272 | 4268 |
| 300 seconds relocation | restarts | 67862 | 36911 | 23979 | 16497 | 13685 |
| 300 seconds search | *Z*1(π) - VNS | 4350 | 4239 | 4247 | 4247 | 4247 |
|  |  |  |  |  |  |  |
| VNS | *Z*1(π) - RH | 4421 | 4280 | 4269 | 4272 | 4268 |
| 300 seconds relocation | restarts | 67877 | 36949 | 24019 | 17493 | 13467 |
| 900 seconds search | *Z*1(π) - VNS | 4350 | 4239 | 4247 | 4247 | 4247 |

The value of *Z*1(π) dropped sharply when increasing from *K* = 2 to *K* = 3, but seemed to level off after *K* = 3. To assure that we were working with the best results possible, we also applied a Fortran implementation of variable neighborhood search to the Bitcoin Alpha network for the same range of *K*. Brusco and Doreian (2019) recently demonstrated the effectiveness of this approach for large sparse signed networks. Unlike rhgsbt, which operates on the full 3783 × 3783 network matrix, the variable neighborhood search algorithm uses only the 24186 edges directly. Because the Bitcoin Alpha network is very sparse (the density is 24186/ (3783)2 = 0.17%), the variable neighborhood search algorithm achieves dramatically more restarts than rhgsbt within the same time limit. This fact, in conjunction with a vigorous search of the neighborhood of the best solution, resulted in significant improvements in the criterion function values, as shown in Table 3. That is, according to the returned values of the criterion function the VNS variant outperforms the RH variant. Although the variable neighborhood search results cannot be verified to be globally optimal, there is considerable stability in the best-found *Z*1(π) values across several different settings. Based on the results in Table 3, it appears that the most appropriate value of the generalized structural balance criterion function is reached at *K* = 3.

***C. Interpretation of the K = 3 blockmodel***

A summary of the three-cluster blockmodel is displayed in Table 5. More specifically, Table 4 reports the number of actors in each cluster, as well as the sum of positive and negative edge weights associated with each block. There is one large cluster (Cluster 1) consisting of *n*1 = 3441 actors, and two appreciably smaller (but nontrivial) clusters of roughly equal size: Cluster 2 (*n*2 = 180) and Cluster 3 (*n*3 = 162). One of the nice properties of these latter two clusters is their within-cluster *purity*. That is, there are only positive (trustworthy) ties within each of these two clusters and no negative (distrust) ties.

Table 4. Three-cluster blockmodel for the Bitcoin Alpha network.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cluster 1 (*n*1 = 3441) | Cluster 2 (*n*2 = 180) | Cluster 3 (*n*3 = 162) |
|  |  |  |  |
| Cluster 1 (*n*1 = 3441) | Positive sum = 42583 | **Positive sum = 335** | **Positive sum = 275** |
|  | **Negative sum** = **2636** | Negative sum = 3785 | Negative sum = 1679 |
|  |  |  |  |
| Cluster 2 (*n*2 = 180) | **Positive sum = 615** | Positive sum = 660 | **Positive sum = 1** |
|  | Negative sum = 823 | **Negative sum** = **0** | Negative sum = 57 |
|  |  |  |  |
| Cluster 3 (*n*3 = 162) | **Positive sum = 374** | **Positive sum = 3** | Positive sum = 356 |
|  | Negative sum = 499 | Negative sum = 316 | **Negative sum** = **0** |

Note – negative sums within clusters and positive sums between clusters are inconsistencies with structural

balance and are highlighted in bold. Summing the values in bold yields *Z*1(π) = 4239.

The off-diagonal blocks in Table 4 reveal that the largest cluster of actors (Cluster 1) exhibits a significant level of distrust of the actors in Cluster 2 and, albeit to a lesser extent, the actors in Cluster 3. The sum of the negative edge weights from actors in Cluster 1 to actors in Cluster 2 is more than 10 times the sum of positive edge weights. Likewise, the sum of the negative edge weights from actors in Cluster 1 to actors in Cluster 3 is more than six times the sum of positive edge weights. The distrust of actors in Cluster 1 toward actors in Clusters 2 and 3 is reciprocated only to a modest extent. For example, the sum of the negative edge weights from actors in Cluster 2 to actors in Cluster 1 is only 1.34 times the sum of positive edge weights, and the sum of the negative edge weights from actors in Cluster 3 to actors in Cluster 1 is only 1.33 times the sum of positive edge weights.

There are very few positive edges between Clusters 2 and 3, indicating that actors in these two clusters do not trust one another. The negative edge weight sums between Clusters 2 and 3 are appreciably larger than the positive sum and, therefore, there is clearly some degree of distrust between the actors in the two clusters. However, it is striking that the sum of the negative edge weights from actors in Cluster 3 to actors in Cluster 2 is more than five times the sum of the negative edge weights from actors in Cluster 2 to actors in Cluster 3. Thus, the level is distrust is not symmetric. Most of the “fraudsters” it seems are in Cluster 2, as there is a high level of distrust from actors in both Clusters 1 and 3 toward the actors in Cluster 2.

***D. References***

Bales, S. (1970). *Personality and interpersonal behavior*. New York: Holt, Rinehart, and Winston.

Brusco, M., & Doreian, P. (2015). A real-coded genetic algorithm for two-mode *KL*-means partitioning with application to homogeneity blockmodeling. *Social Networks*, *41*, 26–35. http://dx.doi.org/10.1016/j.socnet.2014.11.007

Brusco, M. J., & Doreian, P. (2019). Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search. *Social Networks*, *56*, 70-80. <https://doi.org/10.1016/j.socnet.2018.08.007>

Brusco, M., Doreian, P., Mrvar, A., & Steinley, D. (2011). Linking theory, models, and data to understand social network phenomena: Two algorithms for relaxed structural balance partitioning. *Sociological Methods and Research*, *40*, 57-87. https://journals.sagepub.com/doi/10.1177/0049124110384947

Brusco, M., Doreian, P., Mrvar, A., & Steinley, D. (2013). An exact algorithm for blockmodeling of two-mode network data. *Journal of Mathematical Sociology*, *37*, 61-84. https://www.tandfonline.com/doi/abs/10.1080/0022250X.2011.597278

Brusco, M., & Steinley, D. (2010). *K*-balance partitioning: An exact method with application to generalized structural balance and other psychological contexts. *Psychological Methods*, *15*, 145-157. [http://dx.doi.org/10.1037/a0017738](https://psycnet.apa.org/doi/10.1037/a0017738)

Doreian, P., Batagelj, V., & Ferligoj, A. (2005). *Generalized blockmodeling*, Cambridge, UK: Cambridge University Press.

Doreian, P., Kapuscinski, R., Krackhardt, D., & Szczypula, J. (1996). A brief history of balance through time. *Journal of Mathematical Sociology*, *21*, 113-131. doi:10.1080/0022250X.1996.9990176

Doreian, P., & Mrvar, A. (1996). A partitioning approach to structural balance. *Social Networks*, *18*, 149-168. <https://doi.org/10.1016/0378-8733(95)00259-6>

Doreian, P., & Mrvar, A. (2009). Partitioning signed social networks. *Social Networks*, *31*, 1-11. [http://dx.doi.org/10.1016/j.socnet.2008.08.001](https://psycnet.apa.org/doi/10.1016/j.socnet.2008.08.001)

Figueiredo, R., & Moura, G. (2013). Mixed integer programming formulations for clustering problems related to structural balance. *Social Networks*, *35*, 639-651. <https://doi.org/10.1016/j.socnet.2013.09.002>

Galaskiewicz, J. (1985). *Social organization of an urban grants economy*. New York: Academic Press.

Goodwin, V. L., Bowler, W. M., & Whittington, J. L. (2009). A social network perspective on LMX relationships: Accounting for the instrumental value of leader and follower networks. *Journal of Management*, *35*, 954-980. [https://doi.org/10.1177/0149206308321555](https://doi.org/10.1177%2F0149206308321555)

Kumar, S., Spezzano, F., Subrahmanian, V. S., Faloutsos, C. (2016). [Edge weight prediction in weighted signed networks](http://cs.stanford.edu/~srijan/pubs/wsn-icdm16.pdf). *IEEE International Conference on Data Mining (ICDM)*, 221-230.

Lau, D. C., & Liden, R. C. (2008). Antecedents of co-worker trust: Leaders’ blessings. *Journal of Applied Psychology*, *93*, 1130-1138. [http://dx.doi.org/10.1037/0021-9010.93.5.1130](https://psycnet.apa.org/doi/10.1037/0021-9010.93.5.1130)

Lemann, T. B., & Solomon, R. L. (1952). Group characteristics as revealed in sociometric patterns and personality ratings. *Sociometry*, *15*, 7-90. [http://dx.doi.org/10.2307/2785447](https://psycnet.apa.org/doi/10.2307/2785447)

McKinney, J. C. (1948). An educational application of a two-dimensional sociometric test. *Sociometry*, *11*, 356-367. doi: 10.2307/2785196

Newcomb, T. N. (1961). *The acquaintance process*. New York: Holt Rinehart and Winston.,

Nordlie, P. G. (1958). *A longitudinal study of interpersonal attraction in a natural group setting*. Dissertation Abstracts, *19*,1461.

Roethlisberger, F. J., & Dickson, W. J. (1939). *Management and the worker*. Cambridge, MA: Harvard University Press.

Sampson, S. F. (1968). *A novitiate in a period of change: An experimental case study of relationships*. Unpublished Ph.D. dissertation, Department of Sociology, Cornell University, Ithaca, NY.