`r_49`: Facebook Relationship Prediction

JHU Team

2017-09-17

Contents

1 Problem

GM250_seed is an instance of the Graph Matching problem.
In this problem two graphs are given. `G1` and `G2`.
A partial map between nodes of `G1` and `G2` are provided in the train data.
The task is to predict the mapping between the unmapped nodes in the test data.

2 Data

Data for GM250_seed consists of two graphs in raw_data dir:

- `G1.gml`: attributed undirected graph; ~~250~~ 755 nodes; 5138 edges; ~~12~~ 4 features for each node
- `G2.gml`: attributed undirected graph; ~~124~~ 755 nodes; ~~324~~ 5138 edges; ~~12~~ 4 features for each node
- `trainData.csv` contains `G1` nodes and `trainTargets.csv` contains `G2` nodes. Together, they constitute 151 known mappings.
- ~~`testData.csv` contains `G2` nodes, for which the mappings have to be predicted in `testTargets.csv`~~

```r
suppressMessages(library(tidyverse))
suppressMessages(library(igraph))
suppressMessages(library(Matrix))
suppressMessages(library(VN))

g1file <- "http://www.cis.jhu.edu/~parky/D3M/r49/data/raw_data/G1.gml"
g2file <- "http://www.cis.jhu.edu/~parky/D3M/r49/data/raw_data/G2.gml"

g1 <- read_graph(g1file,format = "gml"); summary(g1); is.connected(g1)
```

```
# IGRAPH 3a1d7fc U--- 1000 5521 --
# + attr: id (v/n), label (v/n), nodeID (v/n), f0 (v/n), f1 (v/n),
# | f2 (v/n), f3 (v/n), f4 (v/n), class (v/n)
```

```
# [1] FALSE
```

```r
g2 <- read_graph(g2file,format = "gml"); summary(g2); is.connected(g2)
```

```
# IGRAPH 1e2817c U--- 755 5138 --
# + attr: id (v/n), label (v/n), f0 (v/n), f1 (v/n), f2 (v/n), f3
# | (v/n), f4 (v/n), class (v/n), nodeID (v/n)
# [1] TRUE
```

```r
# find lcc of g1
cl <- igraph::clusters(g1)
g1 <- induced.subgraph(g1, which(cl$membership == which.max(cl$csize)))
summary(g1)
```

```
# IGRAPH 7e0198c U--- 755 5138 --
# + attr: id (v/n), label (v/n), nodeID (v/n), f0 (v/n), f1 (v/n),
# | f2 (v/n), f3 (v/n), f4 (v/n), class (v/n)
```

```r
isomorphic(g1, g2)
```

```
# [1] TRUE
```

```r
# id == nodeID
# revise node ids
V(g1)$id <- 1:vcount(g1)
V(g2)$id <- 1:vcount(g2)
```

```r
train1 <- read_csv("http://www.cis.jhu.edu/~parky/D3M/r49/data/trainData.csv")
train2 <- read_csv("http://www.cis.jhu.edu/~parky/D3M/r49/data/trainTargets.csv")
#test <- read_csv("~/Dropbox/D3M/D3M/r49/solution/baseline/testTargets.csv")
train <- left_join(train1,train2, by="d3mIndex"); train
```

```
# # A tibble: 151 x 5
#    d3mIndex graph.x G1.nodeID graph.y G2.nodeID
#       <int>   <chr>     <int>   <chr>     <int>
#  1        0  G1.gml         7  G2.gml        32
#  2        1  G1.gml       152  G2.gml       433
#  3        2  G1.gml       742  G2.gml       831
#  4        3  G1.gml        12  G2.gml       616
#  5        4  G1.gml       277  G2.gml       515
#  6        5  G1.gml       279  G2.gml       796
#  7        6  G1.gml       430  G2.gml       457
#  8        7  G1.gml        15  G2.gml       680
#  9        8  G1.gml       117  G2.gml       401
# 10        9  G1.gml       389  G2.gml       737
# # ... with 141 more rows
```

```r
# rearrange the graphs so that seeds are the first m vertices
matched.id1 <- match(train$G1.nodeID, V(g1)$nodeID) # 151
perm.g1 <- invPerm(c(matched.id1, (1:vcount(g1))[-matched.id1]))
matched.id2 <- unique(match(train$G2.nodeID, V(g2)$nodeID)) # 145
perm.g2 <- invPerm(c(matched.id2, (1:vcount(g2))[-matched.id2]))
g1.new <- permute.vertices(g1, perm.g1); head(V(g1.new)$nodeID, 10)
```

```
#  [1]   7 152 742  12 277 279 430  15 117 389
```

```r
g2.new <- permute.vertices(g2, perm.g2); head(V(g2.new)$nodeID, 10)
```

```
#  [1]  32 433 831 616 515 796 457 680 401 737
```

```r
g2.sub <- induced.subgraph(g2.new, 1:nrow(train)); summary(g2.sub)
```

```
# IGRAPH 39b70ab U--- 151 215 --
# + attr: id (v/n), label (v/n), f0 (v/n), f1 (v/n), f2 (v/n), f3
# | (v/n), f4 (v/n), class (v/n), nodeID (v/n)
```

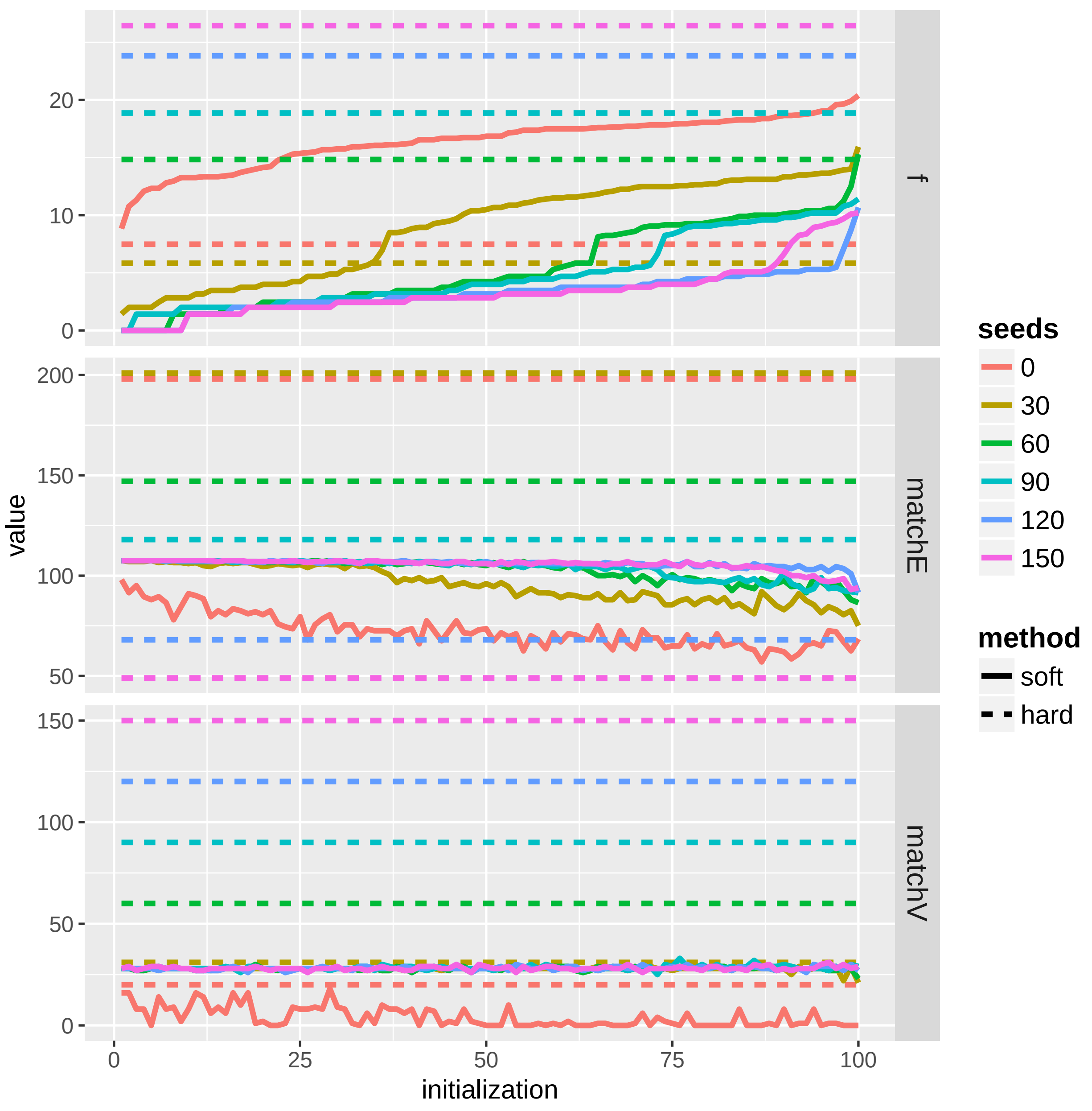
3 Seeded Graph Matching

So, m = 151 correspondence are given.
We will use the first $s = \{0, 30, 60, 90, 120, 150\}$ vertices as seeds and repeat the process 100 times to see the matching performance.

> On Sep 9, 2017, at 12:18 PM, Vince Lyzinski vincelyzinski@gmail.com wrote:
>
> hard seeding enforces the seeds throughout the problem (they can't change), while soft seeding just initializes at the seeds, but allows them to change in the course of the optimization

```
set.seed(12345)

A1 <- as.matrix(g1.new[])
A2 <- as.matrix(g2.new[])
n <- nrow(A1)
m <- nrow(train)
gamma <- 1

nmc <- 100
niter <- 100
svec <- seq(0, 150, by=30)
```

```
method <- "soft"
nmc <- ifelse(method=="soft", 100, 1)
niter <- ifelse(method=="soft", 100, 30)

for (s in svec) {
    cat("Working on s = ", s, "\n")
    mc <- foreach (i=1:nmc) %dopar% {
        ## S is a starting point for softseeding
        if (method=="soft") {
            M <- rsp(n-s,gamma)
            S <- diag(n);
            S[(s+1):n,(s+1):n] <- M
            out <- sgm(A2,A1,0,start=S,pad=0,iteration=niter)
        } else { # "hard"
            S <-matrix(1/(n-s), n-s, n-s)
            out <- sgm(A2,A1,s,start=S,pad=0,iteration=niter)
            if (s > 0) {
                out$corr <- c(1:s,out$corr)
            }
        }

        newA2 <- out$P %*% A1 %*% t(out$P)
        f <- norm(A2[1:m,1:m]-newA2[1:m,1:m], "F")
        matchV <- sum(out$corr[1:m] == 1:m)
        matchE <- sum((A2[1:m,1:m]+newA2[1:m,1:m])>=2) / 2
        c(matchV, matchE, f)
    }
    save(mc,n,m,nmc,s,gamma,file=paste0("mc-r49-s",s,"-nmc",nmc,"-niter",niter,"-",method,".Rbin"))
}
```



value
f
matchE
matchV
initialization
0
25
50
75
100
seeds
0
30
60
90
120
150
method
soft
hard