ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Stable Marriage Problem

Valentin Junet & Samuel Imfeld

Zurich
December 2014

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Valentin Junet

Samuel Imfeld

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

1 Abstract

Online dating or match-making websites are flourishing these days. More and more people rely on their algorithms when searching for their Mr. Right, or Mrs. Right, respectively. Algorithms for match-making are therefore of quite some interest.

The goal of this paper is to discuss the original model described by Gale and Shapley [1962] and advance the model in some sense. Namely we are going to introduce two changes to the model:

1. In the original model every node knows all the other nodes of the opposite gender. In a setting like a database of some match-making site this may be true. But as soon as the number of nodes gets big, it costs a huge amount of computation time to consider all nodes. In reality, information about the nodes of opposite is never complete (this would mean knowing about 3.5 billion people).

2. It is also conceivable that at some point a node might change its opinion about other nodes and rearrange them in his preference rating

It is however not our claim that these changes applied to the model will make it an exact description of reality. Our goal is to study the repercussions on the stability and other significant indicators that show up when applying the modifications.

2 Individual contributions

3 Introduction and Motivations

4 Description of the Model

Gale and Shapley [1962]

5 Implementation

6 Simulation Results and Discussion

7 Summary and Outlook

8 References

References

D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):pp. 9–15, 1962. ISSN 00029890. URL http://www.jstor.org/stable/2312726.

9 Appendix: MATLAB Codes

generateRandom.m

```
function [ m, f ] = generateRandom( n )
%GENERATERANDOM generates random preference matrices
m = zeros(n,n);
f = zeros(n,n);
for i=1:n
    m(i,:) = randperm(n,n);
    f(i,:) = randperm(n,n);

end
```

generatePlane.m

```
function [ mpref,fpref ] = generatePlane( n ,mode, radius)
%GENERATEPLANE generates preference lists for men and women
%   based on a plane where women and men are represented by points
%   they have a limited visibility radius
%   n: number of men and women
%   mode: visibility radius mode, optional argument
%     1 --> const, one constant radius for all nodes
%     2 --> random, a new random radius is generated in each iteration
%           value is between 0.1 and 0.5
%   default mode is const
%   mpref: mens preferences in nxn matrix
```

```
%   fpref: womens preferences in nxn matrix

global verbosity

if (nargin >= 2 && mode == 1)
    assert(nargin==3);
    r = radius;
end
if(nargin < 2)
    mode = 1;
    r = 0.2;%default value
end

% generate random coordinates
% and extend to torus
men = zeros(3,9*n);
rnd = rand(2,n);
men(:,(0*n)+1:1*n)=[(1:n);rnd];
men(:,(1*n)+1:2*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);ones(1,n);zeros(1,n)];
men(:,(2*n)+1:3*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);ones(1,n);ones(1,n)];
men(:,(3*n)+1:4*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);zeros(1,n);ones(1,n)];
men(:,(4*n)+1:5*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);-ones(1,n);ones(1,n)];
men(:,(5*n)+1:6*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);-ones(1,n);zeros(1,n)];
men(:,(6*n)+1:7*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);-ones(1,n);-ones(1,n)];
men(:,(7*n)+1:8*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);zeros(1,n);-ones(1,n)];
men(:,(8*n)+1:9*n)=men(:,(0*n)+1:1*n)+[zeros(1,n);ones(1,n);-ones(1,n)];

women = zeros(3,9*n);
rnd = rand(2,n);
women(:,(0*n)+1:1*n)=[(1:n);rnd];
women(:,(1*n)+1:2*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);ones(1,n);zeros(1,n)];
women(:,(2*n)+1:3*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);ones(1,n);ones(1,n)];
women(:,(3*n)+1:4*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);zeros(1,n);ones(1,n)];
women(:,(4*n)+1:5*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);-ones(1,n);ones(1,n)];
women(:,(5*n)+1:6*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);-ones(1,n);zeros(1,n)];
women(:,(6*n)+1:7*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);-ones(1,n);-ones(1,n)];
women(:,(7*n)+1:8*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);zeros(1,n);-ones(1,n)];
women(:,(8*n)+1:9*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);ones(1,n);-ones(1,n)];

%plotting
if verbosity~=0
    plot(men(2,1:n),men(3,1:n),'o',women(2,1:n),women(3,1:n),'o');
    label1 = cellstr( num2str(women(1,1:n)') );
    label2 = cellstr( num2str(men(1,1:n)') );
    text(women(2,1:n),women(3,1:n),label1);
    text(men(2,1:n),men(3,1:n),label2);
    title('nodes in plane');
    legend('men','women');
end

```

```
d = zeros(2,9*n);
mpref = zeros(n,n);
fpref = zeros(n,n);

for i=1:n
    man = men(:,i);
    for j=1:9*n
        woman = women(:,j);
        d(:,j) = [woman(1,1);norm(man(2:3)-woman(2:3),2)];
    end
    if mode==2
        r = rand*0.4+0.1;
    end
    index = find(d(2,:)<r);
    available = women(:,index);
    sz = size(available,2);
    if sz>n
        available = available(:,1:n);
        sz = n;
    end
    perm = randperm(sz);
    mpref(i,1:sz) = available(1,perm);
end

for i=1:n
    woman = women(:,i);
    for j=1:9*n
        man = men(:,j);
        d(:,j) = [man(1,1);norm(man(2:3)-woman(2:3),2)];
    end
    if mode==2
        r = rand*0.4+0.1;
    end
    index = find(d(2,:)<r);
    available = men(:,index);
    sz = size(available,2);
    if sz>n
        available = available(:,1:n);
        sz = n;
    end
    perm = randperm(sz);
    fpref(i,1:sz) = available(1,perm);
end
end
```

vprintf.m

```
function vprintf(varargin)
% VPRINTF controlled printing
```

```
%
global verbosity
if verbosity~=0
    fprintf(varargin{:});
end
```

makeMatch.m

```
function [ engaged, output ] = makeMatch( m, f )
%makeMatch finds engagements for preferences according to Gale—Shapley ...
    algorithm
%   men an women encoded as integers from 1 to n
%   m ==> preference matrix of the men. Each row corresponds to a man and
%   the elements are the women listed according to his preferences.
%   f ==> preference matrix of the women. Each row corresponds to a woman and
%   the elements are the men listed according to her preferences.
%   Dimensions must be correct, m=nxn, f=nxn.
%   returns:
%   engaged: nx2 Matrix containing matches
%   output: output data —>
%   output(1,1): number of instabilities
%   output(1,2): number of singles
%   output(1,3): number of dumps
%   output(1,4): optimality index
global verbosity
vprintf('mens preferences:');
if verbosity~=0 disp(m); end
vprintf('womens preferences:')
if verbosity~=0 disp(f); end
initialm = m;
initialf = f;
n = size(m,1);
n2 = size(f,1);
assert(n == size(m,2));
assert(n==n2);%make sure dimensions agree
freemen = [(1:n)',ones(n,1)]; %column 1= men; column 2: 1==>man is free, ...
    0==>man isn't free
engaged = zeros(n,2);%column 1=men; column 2=women
dumped=0;%no of dumps
while ~isempty(find(freemen(:,2)==1,1))
    theman = find(freemen(:,2)==1,1); %the first man free on the list
    thegirl = m(theman,1);%his first choice
        if thegirl==0; %"theman" doesn't know any free girls who want him, ...
            he'll be alone :(
            freemen(theman,2)=0;
            engaged(theman,:)=0;
        else
            index = find(engaged(:,2)==thegirl,1);%index of possible fiance ...
                of his first choice
```

```
38              if(isempty(index) ) %"thegirl" is free ==> "theman" will be ...
                    engaged to "thegirl".
39                  if isempty(find(f(thegirl,:)==theman,1))
40                      vprintf('man %d proposed to women %d, she does not know ...
                            him\n', theman, thegirl);
41                      if rand>0.5
42                          engaged(theman,1) = theman;%make new engagement
43                          engaged(theman,2) = thegirl;
44                          vprintf('she accepts\nman %d is engaged to girl ...
                                %d\n', theman, thegirl);
45                          freemen(theman,2) = 0;%man is not free anymore
46                          f(thegirl,:) = [theman, f(thegirl,1:n-1)];
47                          initialf(thegirl,:) = [theman, ...
                                initialf(thegirl,1:n-1)];
48                      else
49                          vprintf('she declines\n');
50                          m(theman,:) = [m(theman,2:n) 0];
51                      end
52                  else
53                      engaged(theman,1) = theman;%make new engagement
54                      engaged(theman,2) = thegirl;
55                      vprintf('man %d is engaged to girl %d\n', theman, thegirl);
56                      freemen(theman,2) = 0;%man is not free anymore
57                  end
58              else %"thegirl" is already engaged ==> check if "thegirl" ...
                    prefers "theman" to her "fiance"
59                  fiance = engaged(index,1);%fiance of first choice
60                  girlprefers = f(thegirl,:);
61                  howgirllikestheman=find(girlprefers==theman,1); %"theman"'s ...
                        number on "thegirl" preferences list
62                  howgirllikesfiance=find(girlprefers==fiance,1); %"fiance"'s ...
                        number on "thegirl" preferences list
63                  if(isempty(howgirllikestheman)) %"thegirl" doesn't know ...
                        "theman" ==> "thegirl" choose beetwen "theman" and her ...
                        "fiance" (the choice is random, with a bigger chance ...
                        for the fiance)
64                      if rand > 0.75
65                          %"thegirl" prefers "theman" ==> actualize
66                          %preference list of "the girl"
67                          f(thegirl,:) = [f(thegirl,1:howgirllikesfiance), ...
                                theman, f(thegirl,howgirllikesfiance+1:n-1)];
68                          initialf(thegirl,:) = ...
                                [initialf(thegirl,1:howgirllikesfiance), ...
                                theman, ...
                                initialf(thegirl,howgirllikesfiance+1:n-1)];
69                      end %if_2
70                  end %if_1
71                  if(find(girlprefers==theman,1)<find(girlprefers==fiance,1)) ...
                        %"thegirl" prefers "theman" ==> change engagement
72                      engaged(theman,1) = theman;%change fiance of the girl
```

```
                    engaged(theman,2) = thegirl;
                    engaged(fiance,1) = 0;%fiance is free again
                    engaged(fiance,2) = 0;
                    vprintf('girl %d dumped man %d for man %d\n', thegirl, ...
                        fiance, theman);
                    dumped=dumped+1;
                    freemen(theman,2) = 0;
                    freemen(fiance,2) = 1;
                else
                    m(theman,:) = [m(theman,2:n) 0];%"thegirl" prefers her ...
                        fiance ==> take "thegirl" out of "theman"'s ...
                        preference list
                end %if_3
            end %if_2
        end %if_1
end %while

if dumped==1
    vprintf('\n%d man has been dumped for another\n\n', dumped);
else
    vprintf('\n%d men have been dumped for others\n\n', dumped);
end %if
single = size(find(engaged(:,2)==0),1);   %number of single men/women
if single==1
    vprintf('There is %d single man/woman\n\n', single);
else
    vprintf('There are %d single men/women\n\n', single);
end %if
[stable, counter] = checkEngagements(engaged,initialm,initialf);%check the ...
    engagements
if (stable)
    vprintf('marriages are stable');
else
    vprintf('marriages are unstable\n');
    if counter==1
        vprintf('there is %d unstable mariage\n', counter);
    else
       vprintf('there are %d unstable mariages\n', counter);
    end %if
end
%optimality index
opt = 0;
for i = 1:n
    he = i;
    she = engaged(he,2);
    if she~=0
        hisindex = find(initialf(she,:)==he,1);
        herindex = find(initialm(he,:)==she,1);
    else
        hisindex = n;
```

```
        herindex = n;
    end
    opt = opt + hisindex + herindex;
end
opt = opt/(2*n*n);
vprintf('optimality index is %1.2f\n',opt);
output = zeros(1,4);
output(1,1) = counter;
output(1,2) = single;
output(1,3) = dumped;
output(1,4) = opt;
end
```

checkEngagements.m

```
function [ stable,counter ] = checkEngagements( engaged, m, f )
%checkEngagements checks whether a set of engagements is stable
%   dimensions must be correct, m=nxn, f=nxn, engaged=nx2
%   men an women encoded as integers from 1 to n
%   returns:
%   stable: true for stable engagements, false otherwise
%   counter: the number of unstable mariages

n = size(m,1);%input size

%invert the engaged matrix such that the new matrix has the index of the
%women on the column one and those of their respective husband in row two
invengaged=zeros(n,2);
copy = engaged(:,[2,1]);
i=1;
while i~=n+1
    index=copy(i,1);
    while index==0 && i~=n%find first index that is nonzero
        i=i+1;
        index=copy(i,1);
    end %while
    if index==0 && i==n
        break;
    end %if
    invengaged(index,:)=copy(i,:);
    i=i+1;
end %while

%main loop
stable=true;
he=1;
counter=0;
while he<=n
    she = engaged(he,2); %she is engaged to he
```

```
35      while she==0 && he~=n   %he is not engaged, so there is no instability ...
            ==> check the next man
36          he = he+1;
37          she = engaged(he,2);
38      end %while
39      if she==0 % ==> he=n is not engaged ==> nothing to check.
40          break;
41      end %if
42
43      hisindex = find(f(she,:)==he,1);
44      herindex = find(m(he,:)==she,1);
45      helikesbetter = m(he,1:herindex);
46      shelikesbetter = f(she,1:hisindex);
47
48      if ~isempty(shelikesbetter) %there is no one on earth she likes better
49          for i=1:size(shelikesbetter) %Loop to check if there is unstability ...
                for the girl
50              guy = shelikesbetter(i); %all the guys she likes better
51              guysgirl = engaged(guy,2); %the guy she is engaged to
52              if guysgirl == 0 && ~isempty(find(m(guy,:)== she,1)) %if this ...
                    guy isn't engaged, then she could be with him ==> unstable, ...
                    unless he doesn't know her.
53                  stable = false;
54                  counter=counter+1;
55                  vprintf('man %d and woman %d like each other better\n', guy, ...
                        she);
56              else
57                  guylikes = m(guy,:);   %the ordered preferences of guy
58                  if (find(guylikes==she,1)<find(guylikes==guysgirl,1)) %if ...
                        guy also likes she better than his wife ==> unstable
59                      stable = false;
60                      counter=counter+1;
61                      vprintf('man %d and woman %d like each other better\n', ...
                            guy, she);
62                  end %if_3
63              end %if_2
64          end %for
65      end %if_1
66
67      %now the other way round
68      if ~isempty(shelikesbetter) %there is no one on earth he likes better
69          for i=1:size(helikesbetter) %Loop to check if there is unstability ...
                for the man
70              girl = helikesbetter(i); %all the girls he likes better
71              girlsguy = invengaged(girl,2); %the girl he is engaged to
72              if girlsguy == 0   %if this girl isn't engaged, then she could ...
                    be with her ==> unstable
73                  stable=false;
74                  vprintf('man %d and woman %d like each other better\n', he, ...
                        girl);
```

```
            else
                girllikes = f(girl,:);   %the ordered preferences of girl
                if (find(girllikes==he,1)<find(girllikes==girlsguy,1))  %if ...
                    guy also likes she better than his wife ==> unstable
                    stable = false;
                    vprintf('man %d and woman %d like each other better\n', ...
                        he, girl);
                end %if_3
            end %if_2
        end %for
    end %if_1

    he=he+1;  %go to the next man
end %while
end
```

simulation.m

```
%simulation

% simulate match making
% n is 2et, t from 1 to 6
% radius is either constant or random
%   when constant, in 0.1:0.05:0.5
% frequency

global verbosity
verbosity = 0;

tmax = 6;
t = 2.^(1:tmax);
r = 0.1:0.05:0.5;
data = zeros(tmax,10,4);

% radius random
for i=1:tmax
   n = t(i);
   [a,b] = generatePlane(n,2);
   [x,y] = makeMatch(a,b);
   data(i,10,:) = y;
end

% radius const
for i=1:tmax
    for j=1:9
        n = t(i);
        radius = r(j);
        [a,b] = generatePlane(n,1,radius);
        [x,y] = makeMatch(a,b);
```

```
        data(i,j,:) = y;
    end
end
% plot optimality index for each radius
hold on
figure(1);
col = hsv(10);
%set(groot,'defaultAxesLineStyleOrder',{'-*',':','o'});
for i=1:10
    plot(1:tmax,data(:,i,4),'color', col(i,:), 'marker', '*','linestyle','--');
    title('optimality index for for different radiuses');

end
arr = ['r','a','n','d','o','m',' ',' ',' ',' ',' ',' ',' '];
xlabel('input size 2^x');
ylabel('optimality index');
legend([num2str(r','radius %1.3f');arr]);
hold off

% plot no of dumps for each radius
figure(2);
for i=1:10
    subplot(3,4,i);
    bar(1:tmax,data(:,i,3));
    xlabel('input size 2^x');
    ylabel('number of dumps');
    ylim([0,100]);
    if i~=10
        title(sprintf('plotting #dumps for radius %1.3f',r(i)));
    else
        title('plotting #dumps for radius random');
    end

end

disp data;
```