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

# 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)];
```

```
47 women(:,(6*n)+1:7*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);-ones(1,n);-ones(1,n)];
48 women(:,(7*n)+1:8*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);zeros(1,n);-ones(1,n)];
49 women(:,(8*n)+1:9*n)=women(:,(0*n)+1:1*n)+[zeros(1,n);ones(1,n);-ones(1,n)];
50
51 %plotting
52 if verbosity~=0
53     plot(men(2,1:n),men(3,1:n),'o',women(2,1:n),women(3,1:n),'o');
54     label1 = cellstr( num2str(women(1,1:n)') );
55     label2 = cellstr( num2str(men(1,1:n)') );
56     text(women(2,1:n),women(3,1:n),label1);
57     text(men(2,1:n),men(3,1:n),label2);
58     title('nodes in plane');
59     legend('men','women');
60 end
61
62 d = zeros(2,9*n);
63 mpref = zeros(n,n);
64 fpref = zeros(n,n);
65
66 for i=1:n
67     man = men(:,i);
68     for j=1:9*n
69         woman = women(:,j);
70         d(:,j) = [woman(1,1);norm(man(2:3)-woman(2:3),2)];
71     end
72     if mode==2
73         r = rand*0.4+0.1;
74     end
75     index = find(d(2,:)<r);
76     available = women(:,index);
77     sz = size(available,2);
78     if sz>n
79         available = available(:,1:n);
80         sz = n;
81     end
82     perm = randperm(sz);
83     mpref(i,1:sz) = available(1,perm);
84 end
85
86 for i=1:n
87     woman = women(:,i);
88     for j=1:9*n
89         man = men(:,j);
90         d(:,j) = [man(1,1);norm(man(2:3)-woman(2:3),2)];
91     end
92     if mode==2
93         r = rand*0.4+0.1;
94     end
95     index = find(d(2,:)<r);
96     available = men(:,index);
```

```
97      sz = size(available,2);
98      if sz>n
99          available = available(:,1:n);
100         sz = n;
101     end
102     perm = randperm(sz);
103     fpref(i,1:sz) = available(1,perm);
104 end
105 end
```

### vprintf.m

```
1  function vprintf(varargin)
2  % VPRINTF controlled printing
3  %
4  global verbosity
5  if verbosity~=0
6      fprintf(varargin{:});
7  end
```

### makeMatch.m

```
1  function [ engaged, output ] = makeMatch( m, f )
2  %makeMatch finds engagements for preferences according to Gale–Shapley ...
       algorithm
3  %   men an women encoded as integers from 1 to n
4  %   m ==> preference matrix of the men. Each row corresponds to a man and
5  %   the elements are the women listed according to his preferences.
6  %   f ==> preference matrix of the women. Each row corresponds to a woman and
7  %   the elements are the men listed according to her preferences.
8  %   Dimensions must be correct, m=nxn, f=nxn.
9  %   returns:
10 %   engaged: nx2 Matrix containing matches
11 %   output: output data -->
12 %   output(1,1): number of instabilities
13 %   output(1,2): number of singles
14 %   output(1,3): number of dumps
15 %   output(1,4): optimality index
16 global verbosity
17 vprintf('mens preferences:');
18 if verbosity~=0 disp(m); end
19 vprintf('womens preferences:')
20 if verbosity~=0 disp(f); end
21 initialm = m;
22 initialf = f;
23 n = size(m,1);
24 n2 = size(f,1);
25 assert(n == size(m,2));
```

```
26 assert(n==n2);%make sure dimensions agree
27 freemen = [(1:n)',ones(n,1)]; %column 1= men; column 2: 1==>man is free, ...
       0==>man isn't free
28 engaged = zeros(n,2);%column 1=men; column 2=women
29 dumped=0;%no of dumps
30 while ~isempty(find(freemen(:,2)==1,1))
31     theman = find(freemen(:,2)==1,1); %the first man free on the list
32     thegirl = m(theman,1);%his first choice
33         if thegirl==0; %"theman" doesn't know any free girls who want him, ...
               he'll be alone :(
34             freemen(theman,2)=0;
35             engaged(theman,:)=0;
36         else
37             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
73                  engaged(theman,2) = thegirl;
74                  engaged(fiance,1) = 0;%fiance is free again
75                  engaged(fiance,2) = 0;
76                  vprintf('girl %d dumped man %d for man %d\n', thegirl, ...
                        fiance, theman);
77                  dumped=dumped+1;
78                  freemen(theman,2) = 0;
79                  freemen(fiance,2) = 1;
80              else
81                  m(theman,:) = [m(theman,2:n) 0];%"thegirl" prefers her ...
                        fiance ==> take "thegirl" out of "theman"'s ...
                        preference list
82              end %if_3
83          end %if_2
84      end %if_1
85  end %while
86
87  if dumped==1
88      vprintf('\n%d man has been dumped for another\n\n', dumped);
89  else
90      vprintf('\n%d men have been dumped for others\n\n', dumped);
91  end %if
92  single = size(find(engaged(:,2)==0),1);   %number of single men/women
93  if single==1
94      vprintf('There is %d single man/woman\n\n', single);
95  else
96      vprintf('There are %d single men/women\n\n', single);
97  end %if
98  [stable, counter] = checkEngagements(engaged,initialm,initialf);%check the ...
        engagements
99  if (stable)
100     vprintf('marriages are stable');
101 else
102     vprintf('marriages are unstable\n');
103     if counter==1
```

```
104         vprintf('there is %d unstable mariage\n', counter);
105     else
106        vprintf('there are %d unstable mariages\n', counter);
107     end %if
108 end
109 %optimality index
110 opt = 0;
111 for i = 1:n
112     he = i;
113     she = engaged(he,2);
114     if she~=0
115         hisindex = find(initialf(she,:)==he,1);
116         herindex = find(initialm(he,:)==she,1);
117     else
118         hisindex = n;
119         herindex = n;
120     end
121     opt = opt + hisindex + herindex;
122 end
123 opt = opt/(2*n*n);
124 vprintf('optimality index is %1.2f\n',opt);
125 output = zeros(1,4);
126 output(1,1) = counter;
127 output(1,2) = single;
128 output(1,3) = dumped;
129 output(1,4) = opt;
130 end
```

### checkEngagements.m

```
1  function [ stable,counter ] = checkEngagements( engaged, m, f )
2  %checkEngagements checks whether a set of engagements is stable
3  %   dimensions must be correct, m=nxn, f=nxn, engaged=nx2
4  %   men an women encoded as integers from 1 to n
5  %   returns:
6  %   stable: true for stable engagements, false otherwise
7  %   counter: the number of unstable mariages
8
9  n = size(m,1);%input size
10
11 %invert the engaged matrix such that the new matrix has the index of the
12 %women on the column one and those of their respective husband in row two
13 invengaged=zeros(n,2);
14 copy = engaged(:,[2,1]);
15 i=1;
16 while i~=n+1
17     index=copy(i,1);
18     while index==0 && i~=n%find first index that is nonzero
19         i=i+1;
```

```
20          index=copy(i,1);
21      end %while
22      if index==0 && i==n
23          break;
24      end %if
25      invengaged(index,:)=copy(i,:);
26      i=i+1;
27  end %while
28
29  %main loop
30  stable=true;
31  he=1;
32  counter=0;
33  while he<=n
34      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);
75              else
76                  girllikes = f(girl,:);   %the ordered preferences of girl
77                  if (find(girllikes==he,1)<find(girllikes==girlsguy,1))  %if ...
                        guy also likes she better than his wife ==> unstable
78                      stable = false;
79                      vprintf('man %d and woman %d like each other better\n', ...
                            he, girl);
80                  end %if_3
81              end %if_2
82          end %for
83      end %if_1
84
85      he=he+1;  %go to the next man
86  end %while
87  end
```

## simulation.m

```
1  %simulation
2
3  % simulate match making
4  % n is 2et, t from 1 to 6
5  % radius is either constant or random
6  %   when constant, in 0.1:0.05:0.5
7  % frequency
8
9  global verbosity
10 verbosity = 0;
11
12 tmax = 6;
13 t = 2.^(1:tmax);
14 r = 0.1:0.05:0.5;
15 data = zeros(tmax,10,4);
16
```

```
% 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

```

```
67 disp data;
```