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):

Stable Marriage Problem

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

Name(s):	**First name(s):**
Junet	Valentin
Imfeld	Samuel

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
Zurich, 10.12.2014

Signature(s)

S. Imfeld

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.

An important result in this concern is the so-called Gale Shapley algorithm proposed by Gale and Shapley [1962]. In 2012, Shapley even received the Nobel Prize in Economics for his work. 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: The incompleteness of information and the possibility of preference changes. The modifications will be discussed further in section 4.2.

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

Code was written by both. In the report, Valentin focused on the chapters 3, 4 and 6.1 whereas Samuel concentrated on chapters 1, 5 and 6.2.

3 Introduction and Motivations

The stable marriage problem is quite well known; it is originally an interesting mathematical problem, but it also has something catchy: Everyone has his own idea of what stability is, how we think it is optimal, and how we would like to be matched, or how we think it should be done. Then there comes this emotionless algorithm that pretends – with the support of some definitions and mathematical proofs – to find all of this for us and to do it as least as good as we would. Of course this is a little exaggerated. But when we think about those internet sites that match people according to some criteria, is't it what we're doing? Letting some numbers find an objective answer to those questions, to our subjective questions?

In order to prevent misunderstandings one must very precisely answer the following questions: What is stability? How can a stable situation be optimal? Of course answering those questions would result in a rather philosophical discussion which is not the goal of our paper. Nevertheless stability and other related indicators (which were going to introduce soon enough) will be the center of our attention. But in order to reduce complexity will only be considering some very abstract and theoretical nodes which, for some practical reasons, will be called men or women.

The key points of our work presented here are to generate random links (or ”friendship”) between men and women (which in the following discussions will be referred to by nodes), to classify those links in a random way in order to get a preference list and to apply the algorithm of Gale and Shapley which matches the nodes according to their preferences. Then we modify some parameters and interpret the results. For a more precise description of the procedure, you may refer to the next section.

3.1 Research Question

We want to analyze the influence that the modifications we are going to apply (see section 4.2) has on the simulation result, i. e. the indicators for stability and optimality (see sections 4.2 and 5.5 for detailed explanation). In particular the questions are:

For stability

- How is the stability influenced when the visibility radius becomes smaller (and therefore also the preference lists)?
- Are there more instabilities if the preference changes are more frequent?

For optimality

- How is the optimality influenced when the visibility radius becomes smaller?
- Is the output less optimal if the preference changes are more frequent?
- Does the optimality depend on the input size?

Here we expect to see the following results: The result is less optimal for a smaller visibility radius because then it is more likely that nodes have no partner (which is considered as suboptimal, see section 4.2). It is also less optimal for a higher preference change rate.

4 Description of the Model

4.1 Gale and Shapley’s algorithm

Let’s first describe the algorithm of Gale and Shapley. The setting is of a very abstract nature. There is a certain number of nodes, half of them are men and the other half are women. The first step is that every node makes a preference list, i. e. a ranking of all the nodes of oppposite sex. Then the algorithm proceeds as follows:

1. single men propose to their favorite women
2. - women accept their favorite suitor or their only suitor

- if a woman has a fiance and a suitor, she chooses the one she likes better
- if a woman rejects a man, then the next woman on the rejected man's list is his new favorite

3. if the single men still have women on their list of preferences, go to step 1

So the algorithm terminates when all nodes have a partner. This is obviously always achieved if the length of all the preference lists corresponds to the number of nodes of opposite sex (see Gale and Shapley [1962, p. 14, Theorem 1]). Gale and Shapley also proved that this algorithm always terminates with an optimal stability. For this purpose they gave precise definitions of stability and optimality:

Definition 1: *An assignment of applicants to colleges will be called unstable if there are two applicants α and β who are assigned to colleges A and B, respectively, although β prefers A to B and A prefers β to α.*[1]

Definition 2: *A stable assignment is called optimal if every applicant is at least as well off under it as under any other assignment.*[2]

These definitions are in terms of colleges and applicants because in their paper they first start with this setting and then go on to the special case of stable marriages. Translated to the words of match-making the definitions would be something like: "A set of marriages is called unstable if under it there are a man and a woman who are not married to each other but prefer each other to their actual mates"[3] and "A stable assignment is called optimal if every man is at least as well off under it as under any other assignment".

4.2 Modifications

We will now present in more detail the modifications we made to this algorithm. Our purpose was to modify and adapt it so that it represents more accurately the reality. Of course it is fundamentally impossible to make a perfect representation of reality through this: If for example a man goes into a bar, he doesn't particularly make a list of the women he prefers and propose to each woman according to his list (or at least this is not systematically observed). However, there are some modifications we can make to make it more realistic. One of these is that a man doesn't necessarily know every women (and vice versa). To improve this point, we modified the list of preferences so that men and women only know each other in a restricted area. For example, a man who lives in China doesn't know a woman who lives in Switzerland, so he won't propose to her (we didn't consider the

[1] Gale and Shapley [1962, p. 10]
[2] Gale and Shapley [1962, p. 10]
[3] Gale and Shapley [1962, p. 11]

case where the man in China could know the Swiss woman though internet or by traveling, he only knows the women "around" him). We also considered the case in which a man knows a woman in his area but she doesn't know him and then this man can propose to her and she might accept or not. The other modification we made is that people, obviously, can change their minds. We modified it so that during the process either only the men, only the women or both can change their mind. These changes of preferences are made each "round" (each time a single man proposes to a woman) with a certain probability which is fixed.

We also introduced some new concepts, namely the number of "dumps" and the "optimality index". The number of "dumps" is the number of times a woman rejected her fiance for another man. This can be seen has an indication of time, the more people "get dumped", the longer it takes to reach a final assignment. The optimality index is defined as the sum over all ranks of the actual partners in the preference list of their respective partners, divided by two times the number of men/women squared (resulting in a normalization, e. g. it only takes values between zero and one). If a node has no partner, it counts as the maximal rank because having no partner is certainly not optimal. The optimality index should somehow give information about the optimality of the outcome, although it does not correspond to definition 2 in section 4.1. A low optimality index corresponds to a rather optimal assignment whereas a high optimality index indicates a not so optimal assignment.

5 Implementation

5.1 Conventions

The nodes (men resp. women) are referenced by integers from one to n where n is the input size (number of men/women).

Accordingly, a preference list is a permutation of the first n integers: The element $\sigma(1)$ is the most preferred node, $\sigma(2)$ the second most preferred and so on. In the case of non-complete information, there are $m < n$ distinct integers followed by zeros to complete the sequence.

5.2 Generating the Preference Lists

5.2.1 Random

For testing purposes we first coded a generator for random preference matrices which basically calls `randperm(n)` n times, resulting in a matrix whose rows represent the preference list of a single node. The generated matrix can be used to simulate the classic case with complete information.

5.2.2 The Plane Algorithm

The question that now arises is the following: How can I generate a preference matrix with non-complete information that represents reality in an appropriate manner? In particular there should be a mechanism for controlling the number of non-zero entries in the preference lists (i. e. controlling the degree of completeness of information). The number of non-zero elements should, however, not be the same for each node (this would mean that each node knows the same fixed number of people, which is clearly not a good representation of real situations). Additionally, situations like 'man x knows woman y but woman y does not know man x' should be considered. To achieve these characteristics we developed the 'generatePlane' algorithm: It is based on the idea that a node only knows his neighbours (e. g. the people in his town or community). Therefore his preference list should consist only of nodes that are closer than a certain distance, which we will call the visibility radius in the further discussions. We realized this by assigning to each node a random position in a two dimensional plane. For simplicity they are distributed in $[0, 1] \times [0, 1]$.

Definition 3: *The visibility radius defines the neighbourhood of a node in* $[0, 1] \times [0, 1]$. *A node can only know other nodes that have a distance smaller than the visibility radius.*

To avoid border effects, the edges are connected (one could view it as a torus). For the actual generation of the preference lists one just has to iterate through all nodes and determine all nodes of opposite sex that are in the disc of visibility radius. We then chose to use a random permutation of these neighbourhood nodes to keep it simple.

A constant visibility radius implies in particular that if node x knows node y, then also node y knows node x. But as we don't want this to be the case all the time we also made an option for a random visibility radius that is updated every step.

5.3 Making Matches

This is the part where the algorithm proposed by Gale and Shapley [1962] actually comes into play. An implementation in MATLAB is shown in listing 1 (The basic ideas for the implementation are taken from RosettaCode [2014] and adapted to MATLAB).

```
function [ engaged, stable ] = makeMatch( m, f )
%makeMatch finds engagements for preferences according to Gale–Shapley ...
    algorithm
...
freemen = [(1:n)',ones(n,1)];
engaged = zeros(n,2);
while ~isempty(find(freemen(:,2)==1,1))
    theman = find(freemen(:,2)==1,1);
    thegirl = m(theman,1);
```

```
9      index = find(engaged(:,2)==thegirl,1);
10     if(isempty(index) )
11         engaged(theman,1) = theman;
12         engaged(theman,2) = thegirl;
13         freemen(theman,2) = 0;
14     else
15         fiance = engaged(index,1);
16         girlprefers = f(thegirl,:);
17         if(find(girlprefers==theman,1)<find(girlprefers==fiance,1))
18             engaged(theman,1) = theman;
19             engaged(theman,2) = thegirl;
20             engaged(fiance,1) = 0;
21             engaged(fiance,2) = 0;
22             freemen(theman,2) = 0;
23             freemen(fiance,2) = 1;
24         else
25             m(theman,:) = [m(theman,2:n) 0];
26         end
27     end
28 end
29 stable = checkEngagements(engaged,m,f);
30 ...
31 end
```

Listing 1 : makeMatch code skeleton

The main structures are the arrays `freemen` and `engaged`. The array `freemen` contains all free men (first column: indices of men, second column: 1 for free, 0 for not free) and `engaged` contains the engagements produced by the algorithm (first column: indices of men, second column: indices of women). The main loop starts in line 6 and continues until all men are in an engagement. Then in the loop the first free man is picked (line 7) and his most preferred girl is determined (line 8). Now one has to distinguish between two cases:

- The girl has no engagement: Everything is fine, the man and the girl are now engaged (lines 11 and 12). Also the man is not free anymore (line 13).
- The girl alread has a fiance: In this case one has to do another distinction:
  - The girl prefers the new man to her fiance: A new engagement is made and the old one is cancelled (lines 18-21). One also has to update `freemen`.
  - The engagements remain unchanged, but the girl is removed from the man's preference list because she is not attainable for him.

In the end the engagements are checked with the `checkEngagements` algorithm described in section 5.4. However, when applying the modifications described in section 4.2 and still using this algorithm one runs into problems (as expected). Therefore we had to adapt the `makeMatch` algorithm to be able to handle the following situations:

- An unknown man proposes to a woman who is not engaged: We decided that in this case the proposing man should have a chance to succeed with his proposal, but this should not always be the case. Therefore we implemented a random decision with a certain probability for accepting (typically around 0.25). This can be seen as a simulation of the real-life situation 'the woman gets to know the unknown man and gets to like him (or not)'.
- An unknown node proposes to a woman who is engaged: We decided to apply the same procedure as above. One could argue that the probability for accepting should be lower because she already has a fiance, but we left that out for the sake of simplicity.
- The preference list of a man is empty but he is not engaged: In this case the man is just left with no partner.

The implementation of the above points will not be discussed here any further. The final algorithm is in the appendix for reference.

Finally we added the preference change functionality: In each iteration step a random decision is made between changing preferences or not, using the probability given in the additional parameter `changerate`. If the answer is positive, the preference list of one randomly chosen node is perturbed a little: A random node in the preference list is chosen and switched with the node one rank lower (for example `[4,2,3,1]` becomes `[4,3,2,1]` after switching nodes 2 and 3). There is also a parameter to determine whether only men, only women or both should change their preferences.

5.4 Checking the Engagements

An important indicator for the later discussion is the stability of the engagements. It can be checked using this algorithm. The main loop is shown in listing 2 (again the basic structure is inspired by RosettaCode [2014])

```
while he<=n
    she = engaged(he,2);
    % make sure that the man has a partner i. e. she is not 0
    % otherwise continue because nothing to check
    ...
    hisindex = find(f(she,:)==he,1);
    herindex = find(m(he,:)==she,1);
    helikesbetter = m(he,1:herindex);
    shelikesbetter = f(she,1:hisindex);
    % check for her
    for i=1:size(shelikesbetter)
        guy = shelikesbetter(i);
```

```
13         guysgirl = engaged(guy,2);
14         guylikes = m(guy,:);
15         if (find(guylikes==she,1)<find(guylikes==guysgirl,1))
16             stable = false;
17         end
18     end
19     % check for him
20     ...
21     he=he+1;
22 end
```

Listing 2 : checkEngagements code fragment

The main loop is an iteration over all men (line 1), but only those who are engaged because having no partner is not considered as an instability (lines 3-5). After having retrieved all the indices and preference lists, one iterates over all nodes that appear before the actual partner in their respective preference lists and checks whether there is an instability (see definition 1 in section 4.1). This results in two for loops, one for the man and one for the woman he is engaged to.

5.5 Simulation

The actual simulation makes iterated calls to `generatePlane` and `makeMatch` to simulate the modified Gale-Shapley Algorithm for different parameter settings. The input parameters are:

- `n`: the input size $n \in \mathbb{N}$
- `mode, radius`: mode determines the choice of the visibility radius (either random or constant), and if the radius is constant then it can be set with the parameter radius $r \in [0, 0.5]$
- `changerate`: the rate at which preference changes are performed $c \in [0, 1]$
- `p`: determines whose preferences are changed $p \in \{0, 0.5, 1\}$

The resulting output variables are:

- no. of instabilities: number of instabilities according to definition 1 in section 4.1
- no. of dumps: number of fiances that have been dumped during match-making, see section 4.2
- no. of singles: number of single nodes that remain after match-making
- optimality index: as defined above, see section 4.2

6 Simulation Results and Discussion

Let's first make a general remark for the interpretation of the results. We labelled our nodes -like in the original algorithm- with man and woman, but a more accurate notation would be "the ones who choose" for the men and "the ones who are chosen" for the women. In the sake of interpretation, this make indeed more sense since, in the real life (and not in our computer), when someone ask a person to go out, the one asking chooses someone of his/her (and the man/woman dilemma again!) taste – preference – and the person who is asked to is chosen and his/her choice will be made according to his/her taste -preference- (at least according to this model). So we could do exactly the same by switching or even mixing both categories and find through our interpretation an even more realistic information about the difference in the results between choosing and being chosen. It is also important to notice that every kind of interpretation with respect to reality is purely hypothetical. This is not a realistic model; it only shows us how this specific way of choosing and being chosen responds to the input parameters.

6.1 Instabilities

6.1.1 Parameters

For the considerations about the instability and the number of singles person, we did the following simulation: Generate a matrix of 2n persons with n in 1, 2,..., 11 and either a random radius or a constant radius r in 0.1, 0.25, 0.4; to make the matches, we chose a probability p – for the change of preferences – in 0, 0.25, 0.5, 0.75, 1, we did this with the change of preferences only for men, only for women and for both. We iterated this simulation 11 times and then took the algebraic mean value as data.

We will start by some considerations about the number of instable (as defined by Shapley and Gale) couple counted at the end of the process. We first note, that if the probability of changing preferences is 0, then there is like in the original algorithm no instability for any kind of matrices. This would mean that the algorithm still holds with a restricted knowledge of the others. If we think of the plane as small or big cities, then the persons who follow this algorithm without changing their mind will be able to find a stable love regardless the size of their respective city... always good to note.

We found out an interesting relation between the visibility radius and the number of instabilities. In general (meaning for every kind and probability in the change of preferences and every size of matrices) we found out that for the big radius 0.25 and 0.4, there is significantly less instability than for the 0.1 radius. Actually, for the two big radius, the mean number of instability is always less than two. For example for 2 12.236 for r=0.1, 0.491 for r=0.25 and 0.218 for r=0.4. This would mean that the people with restricted possibilities who change their mind are more likely to end up in an unstable union than the people who have more choice.

graph instability: Legend: Number of instabilities in function of the number of persons, for a probability in the change of preferences fixed at 100% and for different radius. 11 persons, the mean number of instability (w.r.t. all the probabilities) is

Lets now consider the influence of whether only the men or only the women change their mind on the instability of the marriages. We noticed that for constants radius there is always more instability when the men change their preferences but, as mentioned in last paragraph, the number of instability for r=0.25 and 0.4 is so low that this difference, although present, is negligible. Except randomness, we cannot find an explanation for this in the way we made the changes in the algorithm since the change of preferences are made at the beginning of each main loop regardless if the change is made by a man or a woman. So this means that, up to randomness, the explanation should lay in the way of choosing and making matches. With this algorithm, once a man is matched, theres nothing we can do to change it, so if he changes his mind and the woman he now likes better likes him better too, theres nothing to do about it, it will be unstable, whereas if a woman changes her mind then the man she likes better might propose to her and even if she is engaged she can be with him. This means then that a person should make sure to be able to leave his partner if he changes his mind...not very surprising. However, what is very curious is that if we consider this relation not with a constant radius but rather with a random radius, then -from a certain amount of people (27 opposite is true: there is more instability when the women change their preferences! With our previous considerations it is clear that, up to randomness, the explanation should be find in the construction of the plane. The fact that the radius are random influences the men and the women similarly, so the true difference (and the reason we introduced it) between constant and random radius is that, with the random one, a man can know someone who doesnt know him and propose to her and respectively a woman can be proposed to by someone she doesnt know (in this case, we arbitrarily decided that she accepts 25% of the time). As we can see in the graphs, in this case the undecided women dont produce particularly more instabilities, but the undecided however produce less instability than with a constant radius. This might either be caused by the fact that a random radius can be bigger than 0.1 and as weve seen in this case there is less instabilities, or because if they like better a woman who doesnt know him, then this wont be unstable. We can then suppose that with any radius the most influent factor of instability is the indecision of the men, the one who choses and cant break a union.

We also noticed that in the case in which both, the men and the women changes their minds we observe that the results are approximately the mean between the ones in which only the men and the ones in which only the women change their mind. We can also quickly mention that the bigger is the probability that a change occur, the bigger is the number of instability (for both, men and women changing their mind), so when people are indecisive and change their mind, the marriages are less stable.

6.2 Singles

We want now to consider how this variances of the algorithm influence the number of singles person. However lets first start by mentioning what doesnt have any effects on this number, namely, the change of preferences, more precisely its rate and whether it only men, women or both are undecided. Indeed, for any fixed radius and fixed number of persons in our simulation, we observe the same results by varying the rate of change and the group of people changing their minds. So, by following the algorithm, the indecision of the participants wont increase -or diminish- the number of singles.

graph No of singles: Legend: Number of singles in function of the number of person, for different radius.

The influent factors are then the number of people and the radius. Looking at the graph we can easily noticed, that with the number of people increasing, the number of singles has a normal distribution; we did the kstest proposed by Matlab for all data on the number of singles in function of the number of persons and it test was positive. We can observe that for the small radius 0.1, there is a lot more of singles men and women than for the others, actually for radius 0.4, it is even unusual to find a single pair. For radius 0.4, resp. radius 0.25 from 27 person left! So the more people know each other, the less single there is. The random radius gives something intermediate, there is though more singles than for the radius 0.4 and 0.25 since the radius 0.1 has way more influence on the number of singles., resp. persons there isnt any single So the more people know each other, the less single there is. The random radius gives something intermediate, there is though more singles than for the radius 0.4 and 0.25 since the radius 0.1 has way more influence on the number of singles.

6.3 Optimality Index

6.3.1 Parameters

The input parameters were chosen as follows: The input size is a power of 2. Due to the quadratic dependence of the `generatePlane` algorithm, the highest power evaluated is $2^{10} = 1024$. Originally we wanted to work with basis 10 but we had to change to basis 2 because of performance reasons as well. For the visibility radius we took the discrete values `0.1:0.1:0.5`, and for the preference change rates values in `0:0.5:1`. For each triple of parameters the simulation was performed 10 times.

6.3.2 Effects of preference changes

Surprisingly the expected effect of the preference changes could not be observed at all. The optimality index is about the same for the extreme values 0 and 1 for the change rate, and also for 0.5. For small input sizes there is a small variation of the optimality index (that

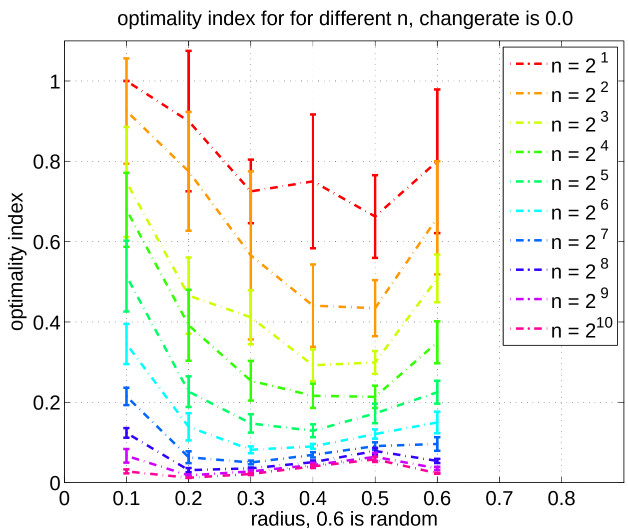


Figure 1: Errorbar plot for the optimality index in dependence of the visibility radius. The different lines correspond to different input sizes n. The preference change rate was set to 0.

seems to be caused more by random effects than an actual correltation). For higher n the plot lines are at constant height (see also section 10.1 in the Appendix for plots).

6.4 Effects of visibility radius

The results of the visibility radius variation do also not look very promising. In figure 1 one can see that for low input sizes (n = 2 to 32) there seems to be a dependence on the visibility radius, namely the optimality index decreases for higher visibility radii (as we expected). But for higher n the effect disappears, and the optimality index even gets higher again for the visibility radius close to 0.5. Note that the points labeled with 0.6 on the x axis correspond to random radius and not visibility radius 0.6.

One way to explain this is that there might be several 'hidden' effects that cause this weird looking behavior. One of them could be that the visibility radius is chosen to be constant for all input sizes. It is now not the same when thousand instead of only ten nodes are placed in the plane, because for more nodes the number of nodes that happen to be in the neighborhood defined by a constant visibility radius is much higher.

6.5 Input size dependency

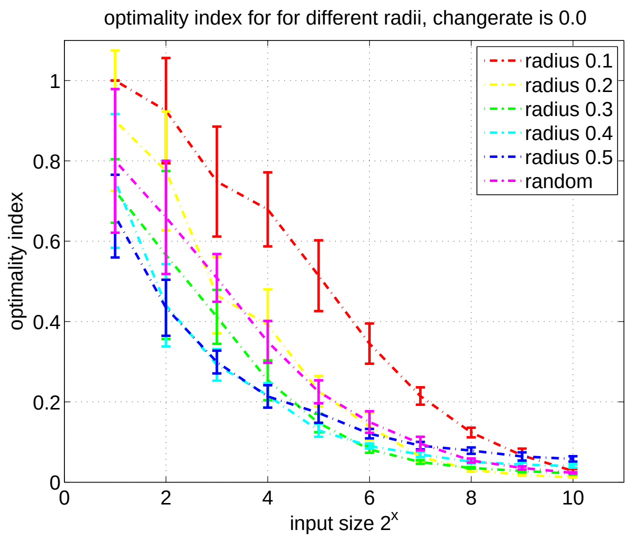


Figure 2: Optimality index errorbar plot in dependence of input size n. Preferences were generated by the `generatePlane` algorithm. Different lines represent different visibility radii.

Looking at figure 2, one can clearly observe that the optimality index gets smaller for increasing input size, meaning that the result is more optimal for higher n. This is a bit counterintuitive at first thought. Why should the result be less optimal for smaller n? One could think that this effect is somehow produced by the modifications that were made to the Gale-Shapley algorithm. But it turns out that the effect can also be observed for randomly generated preference lists using `generateRandom`, see figure 3. For this type of preference lists none of the 'special treatments' in the modified algorithm (e. g. empty preference lists) apply, but the effect can still be observed.

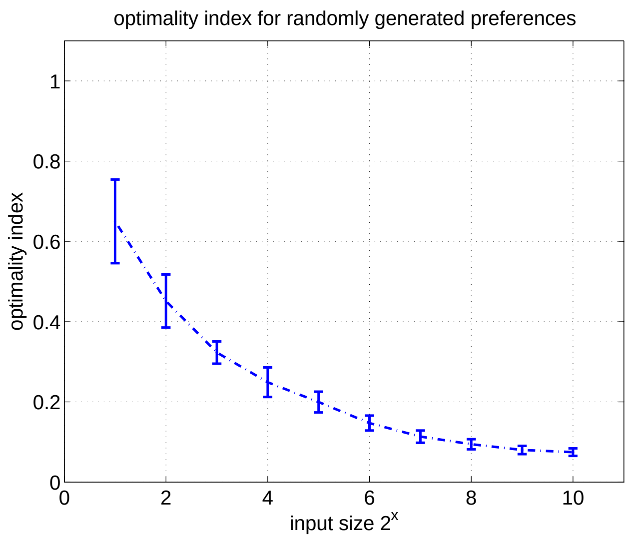


Figure 3: Optimality index errorbar plot in dependence of input size n. Preferences were generated randomly.

7 Summary and Outlook

8 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`.

RosettaCode. Stable marriage problem, 2014. URL `http://rosettacode.org/wiki/Stable_marriage_problem#Python`.

9 Appendix A: 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)];
```

```matlab

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, changerate, p )
%makeMatch finds engagements for preferences according to Gale-Shapley ...
    algorithm
%
%   men an women encoded as integers from 1 to n
%
%   input:
%   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.
%   changerate: rate at which preference changes are performed, e. g. if
%   changerate=0.2 then only in 20% of iterations preferences are changed
%   p: change preferences for men (p=1) / women (p=0.5) / both (p=0.5)
%
%   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

% optional test prints
global verbosity
vprintf('mens preferences:\n');
if verbosity~=0 disp(m); end
vprintf('womens preferences:\n')
if verbosity~=0 disp(f); end
% assign local variables
initialm = m;
initialf = f;
n = size(m,1);
n2 = size(f,1);
% make sure dimensions agree
assert(n == size(m,2));
assert(n==n2);
if nargin > 2
    assert(nargin==4);
    assert(changerate<=1);
    assert(changerate>=0);
    assert(~isempty(find(p==[0,1,0.5],1)));
end
% more local variables
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
acceptrate = 0.75; % rate at which unknown nodes are accepted
% main loop
while ~isempty(find(freemen(:,2)==1,1)) % iterate as long as there are free men
    % preference changes
    if nargin > 2 % only if changerate and p are given
        if rand < changerate % change prefs?
            node = randi(n); % node whose prefs to change
            if rand < p % change for men or women
                pref = nonzeros(m(node,:))';
                len = size(pref,2);
                if len>1
                    k = randi([2,len]); % where in pref to change
                    girl1 = pref(k); % the girl to swap
                    i1 = find(initialm(node,:)==girl1,1); % index of girl1 ...
                        in initialm
                    girl2 = m(node, k-1); % girl to be swapped with
```

```
                i2 = find(initialm(node,:)==girl2,1); % index of girl2 ...
                    in initialm
                initialm(node, i2) = girl1;
                initialm(node, i1) = girl2;
                m(node, i1) = girl2;
                m(node, i1-1) = girl1;
            end
        else
            pref = nonzeros(f(node,:))';
            len = size(pref,2);
            if len>1
                k = randi([2,len]); % where in pref to change
                man1 = pref(k); % the man to swap
                i1 = find(initialf(node,:)==man1,1); % index of man1 in ...
                    initialf
                man2 = f(node, k-1); % man to be swapped with
                i2 = find(initialf(node,:)==man2,1); % index of man2 in ...
                    initialf
                initialf(node, i2) = man1;
                initialf(node, i1) = man2;
                f(node, i1) = man2;
                f(node, i1-1) = man1;
            end
        end %if_2
        vprintf('preferences changed\n');
    end %if_1
end
% +++
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
        if(isempty(index) ) % thegirl is free -> theman will be engaged ...
            to thegirl
            if isempty(find(f(thegirl,:)==theman,1))
                vprintf('man %d proposed to women %d, she does not know ...
                    him\n', theman, thegirl);
                if rand>acceptrate % man accepts with a certain rate
                    engaged(theman,1) = theman; % make new engagement
                    engaged(theman,2) = thegirl;
                    vprintf('she accepts\nman %d is engaged to girl ...
                        %d\n', theman, thegirl);
                    freemen(theman,2) = 0; % man is not free anymore
                    f(thegirl,:) = [theman, f(thegirl,1:n-1)]; % update ...
                        preferences
```

```
                    initialf(thegirl,:) = [theman, ...
                        initialf(thegirl,1:n-1)]; % also in initial ...
                        matrix (will be used for checking)
                else
                    vprintf('she declines\n');
                    m(theman,:) = [m(theman,2:n) 0]; % make pref list ...
                        of theman smaller
                end % if_4
            else
                engaged(theman,1) = theman; % make new engagement
                engaged(theman,2) = thegirl;
                vprintf('man %d is engaged to girl %d\n', theman, thegirl);
                freemen(theman,2) = 0; % man is not free anymore
            end % if_3
        else % thegirl is already engaged -> check if thegirl prefers ...
            theman to her fiance
            fiance = engaged(index,1); % her fiance
            girlprefers = f(thegirl,:); % pref list of thegirl
            howgirllikestheman=find(girlprefers==theman,1); % themans ...
                index on thegirls preferences list
            howgirllikesfiance=find(girlprefers==fiance,1); % fiances ...
                index on thegirls preferences list
            if(isempty(howgirllikestheman)) % thegirl doesn't know ...
                theman -> thegirl accepts with a certain rate
                if rand > 0.75
                    % thegirl prefers theman -> update pref list
                    f(thegirl,:) = [f(thegirl,1:howgirllikesfiance), ...
                        theman, f(thegirl,howgirllikesfiance+1:n-1)];
                    initialf(thegirl,:) = ...
                        [initialf(thegirl,1:howgirllikesfiance), ...
                        theman, ...
                        initialf(thegirl,howgirllikesfiance+1:n-1)]; % ...
                        also initial
                end % if_4
            end % if_3
            if(find(girlprefers==theman,1)<find(girlprefers==fiance,1)) ...
                % thegirl prefers theman ->change engagement
                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 themans preference list
            end % if_3
```

```
            end % if_2
        end % if_1
end % while
% result printing (suppressed if verbositiy set to 0)
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 nodes
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\n');
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_2
end % if
% calculate 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);
% set output
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
%
%   men an women encoded as integers from 1 to n
%
%   input:
%   engaged: engagement matrix
%   m,f: preference matrices
%
%   dimensions must be correct, m=nxn, f=nxn, engaged=nx2
%
%   returns:
%   stable: true for stable engagements, false otherwise
%   counter: the number of unstable mariages

n = size(m,1); % input size
% reverse the engaged matrix such that the new matrix has the index of the
% women on the column one and those of their respective husbands 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
% assign local variables
stable=true;
he=1;
counter=0;
inst = [0,0];
% main loop
while he<=n
    she = engaged(he,2); % she is engaged to he
    while (she==0 && he~=n) % he is not engaged, so there is no instability ... -> check the next man
        he = he+1;
        she = engaged(he,2);
    end % while
    if she==0 % -> he=n is not engaged, nothing to check.
```

```
            break;
        end %if
        % get indexes in pref lists
        hisindex = find(f(she,:)==he,1);
        herindex = find(m(he,:)==she,1);
        helikesbetter = m(he,1:herindex);
        shelikesbetter = f(she,1:hisindex);
        % check for her
        if ~isempty(shelikesbetter) % there is no one on earth she likes better
            for i=1:size(shelikesbetter) % loop to check if there is ...
                unstability for the girl
                guy = shelikesbetter(i); % all the guys she likes better
                guysgirl = engaged(guy,2); % the guy she is engaged to
                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.
                   stable = false;
                   vprintf('man %d and woman %d like each other better\n', guy, ...
                       she);
                   inst = [guy,she;inst];

                else
                    guylikes = m(guy,:); % the ordered preferences of guy
                    if (find(guylikes==she,1)<find(guylikes==guysgirl,1)) % if ...
                        guy also likes she better than his wife -> unstable
                        stable = false;
                        vprintf('man %d and woman %d like each other better\n', ...
                            guy, she);
                        inst = [guy,she;inst];
                    end % if_3
                end % if_2
            end % for
        end % if_1
        % now the other way round, check for him
        if ~isempty(helikesbetter) % there is no one on earth he likes better
            for i=1:size(helikesbetter) % loop to check if there is unstability ...
                for the man
                girl = helikesbetter(i); % all the girls he likes better
                girlsguy = invengaged(girl,2); % the girl he is engaged to
                if girlsguy == 0 && ~isempty(find(f(girl,:)== he,1))% if this ...
                    girl isn't engaged, then she could be with her -> unstable
                    stable = false;
                    vprintf('man %d and woman %d like each other better\n', he, ...
                        girl);
                    inst = [he,girl;inst];
                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);
                    inst = [he,girl;inst];
                end % if_3
            end % if_2
        end % for
    end % if_1

    he=he+1; % go to the next man
end % while
% delete duplicate instabilities
inst = unique(inst, 'rows');
counter = size(inst,1)-1;
end
```

simulation.m

```
function [ data ] = simulation( saveit )
%simulation perform simulation
%
%   input:
%   saveit: if 1 data is saved, if 0 not
%
%   returns:
%   data: simulation data


%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;

assert(~isempty(find(saveit==[0,1],1)));
tmax = 6;
t = 2.^(1:tmax);
r = 0.1:0.05:0.5;
data = zeros(tmax,10,4);
seed = rng;
if saveit==1
    dirname = sprintf('data/%s',datestr(now,'yyyy_mm_dd_HH_MM_SS'));
    mkdir(dirname);
end

```

```
% 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
handle = 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]);
if saveit==1
    saveas(handle,sprintf('%s/figure_1.pdf', dirname));
end
hold off

% plot no of dumps for each radius
handle = 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
if saveit==1
    saveas(handle,sprintf('%s/figure_2.pdf', dirname));
end
% saving
if (saveit==1)
    save(sprintf('%s/data.mat',dirname),'data','seed');
end

end
```

simulation2.m

```
function [ data ] = simulation2( saveit )
%simulation perform simulation
%
%   input:
%   saveit: if 1 data is saved, if 0 not
%
%   returns:
%   data: simulation data


%simulation

% simulate match making
% n is 2et, t from 1 to 6
% radius is either constant or random
%   when constant, r=0.4
%makeMatch with preference changes for men, women, men/women and
%probability of changing in p=0.1:0.1:0.9

global verbosity
verbosity = 0;

assert(~isempty(find(saveit==[0,1],1)));
tmax = 11;
t = 2.^(1:tmax);
p = 0.0:0.25:1.0;
pmax=size(p,2);
m=12; %number of different makeMatch called in simulation2
data = zeros(tmax,m*pmax,4);
r= 0.1:0.15:0.4;
seed = rng;
if saveit==1
    dirname = sprintf('data/%s',datestr(now,'yyyy_mm_dd_HH_MM_SS'));
    mkdir(dirname);
end
```

```
% radius random
for i=1:tmax
    n = t(i);
    [a,b] = generatePlane(n,2);
    fprintf('Plane with rand radius, size %d is generated\n', n);
    for j=1:pmax
       [x1,y1] = makeMatch(a,b, p(j),1);% x: engagement matrix; y(1): ...
           #unstable mariage, y(2): #single men/women, y(3): #dumps, y(4): ...
           optimality index
       [x2,y2] = makeMatch(a,b, p(j),0);
       [x3,y3] = makeMatch(a,b, p(j),0.5);
       data(i,j,:) = y1;
       data(i,pmax+j,:)=y2;
       data(i,2*pmax+j,:)=y3;
    end
end

%constant radius in r=[0.1,0.25,0.4]
for i=1:tmax
    n = t(i);
    [a,b] = generatePlane(n,1, r(1));
    fprintf('Plane with radius r=0.1, size %d is generated\n', n);
    for j=1:pmax
       [v1,w1] = makeMatch(a,b, p(j),1);% x: engagement matrix; y(1): ...
           #unstable mariage, y(2): #single men/women, y(3): #dumps, y(4): ...
           optimality index
       [v2,w2] = makeMatch(a,b, p(j),0);
       [v3,w3] = makeMatch(a,b, p(j),0.5);
       data(i,3*pmax+j,:) = w1;
       data(i,4*pmax+j,:) = w2;
       data(i,5*pmax+j,:) = w3;
    end
end

for i=1:tmax
    n = t(i);
    [a,b] = generatePlane(n,1, r(2));
    fprintf('Plane with radius r=0.25, size %d is generated\n', n);
    for j=1:pmax
       [v1,w1] = makeMatch(a,b, p(j),1);% x: engagement matrix; y(1): ...
           #unstable mariage, y(2): #single men/women, y(3): #dumps, y(4): ...
           optimality index
       [v2,w2] = makeMatch(a,b, p(j),0);
       [v3,w3] = makeMatch(a,b, p(j),0.5);
       data(i,6*pmax+j,:) = w1;
       data(i,7*pmax+j,:) = w2;
       data(i,8*pmax+j,:) = w3;
    end
end
```

```
for i=1:tmax
    n = t(i);
    [a,b] = generatePlane(n,1, r(3));
    fprintf('Plane with radius r=0.4, size %d is generated\n', n);
    for j=1:pmax
       [v1,w1] = makeMatch(a,b, p(j),1);% x: engagement matrix; y(1): ...
           #unstable mariage, y(2): #single men/women, y(3): #dumps, y(4): ...
           optimality index
       [v2,w2] = makeMatch(a,b, p(j),0);
       [v3,w3] = makeMatch(a,b, p(j),0.5);
       data(i,9*pmax+j,:) = w1;
       data(i,10*pmax+j,:) = w2;
       data(i,11*pmax+j,:) = w3;
    end
end




% saving
if (saveit==1)
    save(sprintf('%s/data.mat',dirname),'data','seed');
end

end
```

simulation3.m

```
function [ data ] = simulation3( saveit , tmax )
%SIMULATION3 perform simulation
%
%   simfeld
%
%   input:
%   saveit: if 1 data is saved, if 0 not
%   tmax: maximal exponent form input size (base 2)
%
%   returns:
%   data: simulation data

% simulate match making
% n is 2^t, t from 1 to tmax
% radius is either constant or random
%   when constant, in 0.1:rstep:0.5
% frequency changerate 0:fstep:1
% in makeMatch argument list p=0.5 -> change pref for both men and women

global verbosity
```

```
verbosity = 0;

assert(~isempty(find(saveit==[0,1],1)));
if nargin < 2
    tmax = 6;
end
t = 2.^(1:tmax);
rstep = 0.1;
r = 0.1:rstep:0.5;
fstep = 0.5;
f = 0:fstep:1;
sizef = size(f,2);
sizer = size(r,2)+1;
m = 10; % number of iterations
data = zeros(tmax,sizer,sizef,m,4);
% data dimensions:
% 1 input size n, tmax
% 2 radius, sizer
% 3 frequency, sizef
% 4 iterations, m
% 5 output values, 4
seed = rng;
disp(seed);
if saveit==1
    dirname = sprintf('data/%s',datestr(now,'yyyy_mm_dd_HH_MM_SS'));
    mkdir(dirname);
end
tic
fprintf('simulating for radius random\n');
% radius random
for i=1:tmax
    n = t(i);
    for k=1:sizef
        freq = f(k);
        for l=1:m
            %[a,b] = generatePlane(n,2);
            [a,b] = generateRandom(n);
            fprintf('.');
            [x,y] = makeMatch(a,b,freq,0.5);
            data(i,sizer,k,l,:) = y;
        end
        fprintf('\n');
    end
    fprintf('n = %4d complete after %5.1f\n', n, toc);
end
if 1==0
fprintf('simuation for radius const\n')
% radius const
for i=1:tmax
    n = t(i);
```

```
    for j=1:sizer-1
        radius = r(j);
        for k=1:sizef
            freq = f(k);
            for l=1:m
                %[a,b] = generatePlane(n,1,radius);
                [a,b] = generateRandom(n);
                fprintf('.');
                [x,y] = makeMatch(a,b,freq,0.5);
                data(i,j,k,l,:) = y;
            end
            fprintf('\n');
        end
        fprintf('radius %1.1f complete\n', radius);
    end
    fprintf('n = %4d complete after %5.1f\n', n, toc);
end
end
% plotting
plot3(data);
% saving
if (saveit==1)
    save(sprintf('%s/data.mat',dirname),'data','seed');
end

end
```

iteratesimulation.m

```
function [data, dataplot] = iteratesimulation(saveit, n)
    global verbosity
    verbosity=0;

    %!!!tmax, m, p, t, pmax must correspond to simulation2!!!
    assert(~isempty(find(saveit==[0,1],1)));
    seed = rng;
    tmax=11;
    m=12; %number of makeMatch called in simulation2
    p = 0.0:0.25:1.0;
    pmax=size(p,2);
    t = 2.^(1:tmax);
    data = zeros(tmax,m*pmax,4,n);
    dataplot=zeros(tmax,m*pmax,4);
    if saveit==1
        dirname = sprintf('data/%s',datestr(now,'yyyy_mm_dd_HH_MM_SS'));
        mkdir(dirname);
    end
    for i=1:n
        data(:,:,:,i)=simulation2(0);
```

```
        dataplot=dataplot+data(:,:,:,i);
        fprintf('iteration %d is done\n', i);
    end
    dataplot=dataplot/(n*1.);

% plot no of unstability for each probability, with change preferences for
% men, with random radius
hold on
handle = figure(1);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,1:pmax,1));
        if i==1
            xlabel('probability (man)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,pmax+1:2*pmax,1));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,2*pmax+1:3*pmax,1));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end


end
if saveit==1
    saveas(handle,sprintf('%s/(1)instability with random radius.pdf', ...
        dirname));
end
hold off

```

```
% plot no of unstability for each probability, with change preferences for
% men, with constant radius=0.1
hold on
handle = figure(2);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,3*pmax+1:4*pmax,1));
        if i==1
            xlabel('probability (man)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,4*pmax+1:5*pmax,1));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,5*pmax+1:6*pmax,1));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end
end
if saveit==1
    saveas(handle,sprintf('%s/(2)instability with constant radius=0.1.pdf', ...
        dirname));
end
hold off

% plot no of unstability for each probability, with change preferences for
% men, with constant radius=0.25
hold on
handle = figure(3);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
```

```
        bar(p,dataplot(i,6*pmax+1:7*pmax,1));
        if i==1
            xlabel('probability (man)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,7*pmax+1:8*pmax,1));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,8*pmax+1:9*pmax,1));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end
end
if saveit==1
    saveas(handle,sprintf('%s/(3)instability with constant ...
        radius=0.25.pdf', dirname));
end
hold off


% plot no of unstability for each probability, with change preferences for
% men, with constant radius=0.4
hold on
handle = figure(4);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,9*pmax+1:10*pmax,1));
        if i==1
            xlabel('probability (man)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
```

```
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,10*pmax+1:11*pmax,1));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,11*pmax+1:12*pmax,1));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# instabilities');
        end
        ylim([0,10]);
        xlim([0,1]);
    end
end

if saveit==1
    saveas(handle,sprintf('%s/(4)instability with constant radius=0.4.pdf', ...
        dirname));
end
hold off



% plot no of dumps for each probability, with change preferences for
% men, with random radius
hold on
handle = figure(5);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,1:pmax,3));
        if i==1
            xlabel('probability (man)');
            ylabel('# dumps');
        end
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
```

```
        bar(p,dataplot(i-tmax,pmax+1:2*pmax,3));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,2*pmax+1:3*pmax,3));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end


end
if saveit==1
    saveas(handle,sprintf('%s/(5)dumps with random radius.pdf', dirname));
end
hold off

% plot no of dumps for each probability, with change preferences for
% men, with constant radius=0.1
hold on
handle = figure(6);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,3*pmax+1:4*pmax,3));
        if i==1
            xlabel('probability (man)');
            ylabel('# dumps');
        end
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,4*pmax+1:5*pmax,3));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end

    if i>2*tmax
```

```
        bar(p,dataplot(i-2*tmax,5*pmax+1:6*pmax,3));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end
end
if saveit==1
    saveas(handle,sprintf('%s/(6)dumps with constant radius=0.1.pdf', ...
        dirname));
end
hold off

% plot no of dumps for each probability, with change preferences for
% men, with constant radius=0.25
hold on
handle = figure(7);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,6*pmax+1:7*pmax,3));
        if i==1
            xlabel('probability (man)');
            ylabel('# dumps');
        end
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,7*pmax+1:8*pmax,3));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,8*pmax+1:9*pmax,3));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end
end
if saveit==1
    saveas(handle,sprintf('%s/(7)dumps with constant radius=0.25.pdf', ...
```

```
        dirname));
end
hold off


% plot no of dumps for each probability, with change preferences for
% men, with constant radius=0.4
hold on
handle = figure(8);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,9*pmax+1:10*pmax,3));
        if i==1
            xlabel('probability (man)');
            ylabel('# dumps');
        end
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,10*pmax+1:11*pmax,3));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,11*pmax+1:12*pmax,3));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# dumps');
        end
        xlim([0,1]);
    end
end

if saveit==1
    saveas(handle,sprintf('%s/(8)dumps with constant radius=0.4.pdf', ...
        dirname));
end
hold off

% plot no of single men/women for each probability, with change preferences for
% men, with random radius
hold on
handle = figure(9);
```

```
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,1:pmax,2));
        if i==1
            xlabel('probability (man)');
            ylabel('# singles');
        end
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,pmax+1:2*pmax,2));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# singles');
        end
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,2*pmax+1:3*pmax,2));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# singles');
        end
        xlim([0,1]);
    end


end
if saveit==1
    saveas(handle,sprintf('%s/(9) singles with random radius.pdf', dirname));
end
hold off

% plot no of single for each probability, with change preferences for
% men, with constant radius=0.1
hold on
handle = figure(10);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,3*pmax+1:4*pmax,2));
        if i==1
            xlabel('probability (man)');
            ylabel('# singles');
        end
        xlim([0,1]);
```

```
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,4*pmax+1:5*pmax,2));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# singles');
        end
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,5*pmax+1:6*pmax,2));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# singles');
        end
        xlim([0,1]);
    end
end
if saveit==1
    saveas(handle,sprintf('%s/(10)singles with constant radius=0.1.pdf', ...
        dirname));
end
hold off

% plot no of single for each probability, with change preferences for
% men, with constant radius=0.25
hold on
handle = figure(11);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,6*pmax+1:7*pmax,2));
        if i==1
            xlabel('probability (man)');
            ylabel('# singles');
        end
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,7*pmax+1:8*pmax,2));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# singles');
        end
        xlim([0,1]);
```

```
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,8*pmax+1:9*pmax,2));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# singles');
        end
        xlim([0,1]);
    end
end
if saveit==1
    saveas(handle,sprintf('%s/(11)singles with constant radius=0.25.pdf', ...
        dirname));
end
hold off


% plot no of single for each probability, with change preferences for
% men, with constant radius=0.4
hold on
handle = figure(12);
for i=1:3*tmax
    subplot(3,tmax,i);
    if i<=tmax
        bar(p,dataplot(i,9*pmax+1:10*pmax,2));
        if i==1
            xlabel('probability (man)');
            ylabel('# singles');
        end
        xlim([0,1]);
        title(sprintf('input size %1.0f',t(i)));
    end

    if i<=2*tmax && i>tmax
        bar(p,dataplot(i-tmax,10*pmax+1:11*pmax,2));
        if i==tmax+1
            xlabel('probability (women)');
            ylabel('# singles');
        end
        xlim([0,1]);
    end

    if i>2*tmax
        bar(p,dataplot(i-2*tmax,11*pmax+1:12*pmax,2));
        if i==2*tmax+1
            xlabel('probability (men/women)');
            ylabel('# singles');
        end
        xlim([0,1]);
```

```
    end
end

if saveit==1
    saveas(handle,sprintf('%s/(12)singles with constant radius=0.4.pdf', ...
        dirname));
end
hold off

%plot no of single over no of person with  random radius
hold on
handle = figure(13);
for i=1:3*pmax
    subplot(3,pmax,i);
    if i<=pmax
        bar(1:tmax,dataplot(:,i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (man) %1.3f %', p(i)));
    end

    if i<=2*pmax && i>pmax
        bar(1:tmax,dataplot(:,i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (women) %1.3f %', p(i-pmax)));
    end

    if i>2*pmax
        bar(1:tmax,dataplot(:,i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (man/women) %1.3f %', p(i-2*pmax)));
    end

end

if saveit==1
    saveas(handle,sprintf('%s/(13)singles over person with random ...
        radius.pdf', dirname));
end
hold off



%plot no of single over no of person with radius=0.1
hold on
handle = figure(14);
for i=1:3*pmax
    subplot(3,pmax,i);
```

```
    if i<=pmax
        bar(1:tmax,dataplot(:,3*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (man) %1.3f %', p(i)));
    end

    if i<=2*pmax && i>pmax
        bar(1:tmax,dataplot(:,3*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (women) %1.3f %', p(i-pmax)));
    end

    if i>2*pmax
        bar(1:tmax,dataplot(:,3*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (man/women) %1.3f %', p(i-2*pmax)));
    end

end

if saveit==1
    saveas(handle,sprintf('%s/(14)singles over person with radius=0.1.pdf', ...
        dirname));
end
hold off



%plot no of single over no of person with radius=0.25
hold on
handle = figure(15);
for i=1:3*pmax
    subplot(3,pmax,i);
    if i<=pmax
        bar(1:tmax,dataplot(:,6*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (man) %1.3f %', p(i)));
    end

    if i<=2*pmax && i>pmax
        bar(1:tmax,dataplot(:,6*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (women) %1.3f %', p(i-pmax)));
    end

```

```
    if i>2*pmax
        bar(1:tmax,dataplot(:,6*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (man/women) %1.3f %', p(i-2*pmax)));
    end

end

if saveit==1
    saveas(handle,sprintf('%s/(15)singles over person with ...
        radius=0.25.pdf', dirname));
end
hold off

%plot no of single over no of person with radius=0.4
hold on
handle = figure(16);
for i=1:3*pmax
    subplot(3,pmax,i);
    if i<=pmax
        bar(1:tmax,dataplot(:,9*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (man) %1.3f %', p(i)));
    end

    if i<=2*pmax && i>pmax
        bar(1:tmax,dataplot(:,9*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (women) %1.3f %', p(i-pmax)));
    end

    if i>2*pmax
        bar(1:tmax,dataplot(:,9*pmax+i,2));
        xlabel('input size 10^x');
        ylabel('# singles');
        title(sprintf('probability (men/women) %1.3f %', p(i-2*pmax)));
    end

end

if saveit==1
    saveas(handle,sprintf('%s/(16)singles over person with radius=0.4.pdf', ...
        dirname));
end
hold off
```

```

% saving
if (saveit==1)
    save(sprintf('%s/data.mat',dirname),'data','dataplot','seed');
end

end
```

plot3.m

```
function [  ] = plot3( data )
%PLOT3 plotting for optimality index analysis
%
%   simfeld
if 1==0
dirname = sprintf('data/%s',datestr(now,'yyyy_mm_dd_HH_MM_SS'));
mkdir(dirname);
% define arrays
sizer = size(data,2);
r=0.1:0.1:0.5;
f=0:0.5:1;
sizef=size(f,2);
tmax = size(data,1);
t=1:tmax;

% x-axis n, diff radius, plots freq
col = hsv(sizer);
for j=1:sizef
    freq = f(j);
    handle = figure(j);
    set(gca,'FontSize',16);
    hold on
    for i=1:sizer
        mm = squeeze(mean(data(:,i,j,:,4),4));
        st = squeeze(std(data(:,i,j,:,4),0,4));
        %plot(1:tmax,data(:,i,1,1,4),'color', col(i,:), 'marker', ...
            '*','linestyle','--');
        errorbar(1:tmax,mm,st,'color', col(i,:), 'marker', ...
            '.','linestyle','-.', 'LineWidth', 2);
    end
    hold off
    box on
    grid on
    title(sprintf('optimality index for for different radii, changerate is ...
        %1.1f',freq));
    arr = ['r','a','n','d','o','m',' ',' ',' ',' '];
    xlabel('input size 2^x');
    ylabel('optimality index');
    ylim([0,1.1]);
```

```
    xlim([0,11]);
    legend([num2str(r','radius %1.1f');arr]);
    saveas(handle,sprintf('%s/figure_%d.pdf', dirname, j));
end

% x-axis radius, diff n, plots freq
col = hsv(tmax);
for j=1:sizef
    freq = f(j);
    handle = figure(j+3);
    set(gca,'FontSize',16);
    hold on
    for i=1:tmax
        mm = squeeze(mean(data(i,:,j,:,4),4));
        st = squeeze(std(data(i,:,j,:,4),0,4));
        %plot(1:tmax,data(:,i,1,1,4),'color', col(i,:), 'marker', ...
            '*','linestyle','--');
        errorbar(0.1:0.1:0.6,mm,st,'color', col(i,:), 'marker', ...
            '.','linestyle','-.', 'LineWidth', 2);
    end
    hold off
    box on
    grid on
    title(sprintf('optimality index for for different n, changerate is ...
        %1.1f',freq));
    %arr = ['r','a','n','d','o','m',' ',' ',' ',' ',' ',' '];
    xlabel('radius, 0.6 is random');
    ylabel('optimality index');
    ylim([0,1.1]);
    xlim([0,0.9])
    legend(num2str(t','n = 2^{%2d}'));
    saveas(handle,sprintf('%s/figure_%d.pdf', dirname, j+3));
end

% x-axis freq, diff n, plots radius
col = hsv(tmax);
for j=1:sizer
    if j~= sizer radius = r(j); end
    handle = figure(j+6);
    set(gca,'FontSize',16);
    hold on
    for i=1:tmax
        mm = squeeze(mean(data(i,j,:,:,4),4));
        st = squeeze(std(data(i,j,:,:,4),0,4));
        %plot(1:tmax,data(:,i,1,1,4),'color', col(i,:), 'marker', ...
            '*','linestyle','--');
        errorbar(f,mm,st,'color', col(i,:), 'marker', '.','linestyle','-.', ...
            'LineWidth', 2);
    end
    hold off
```

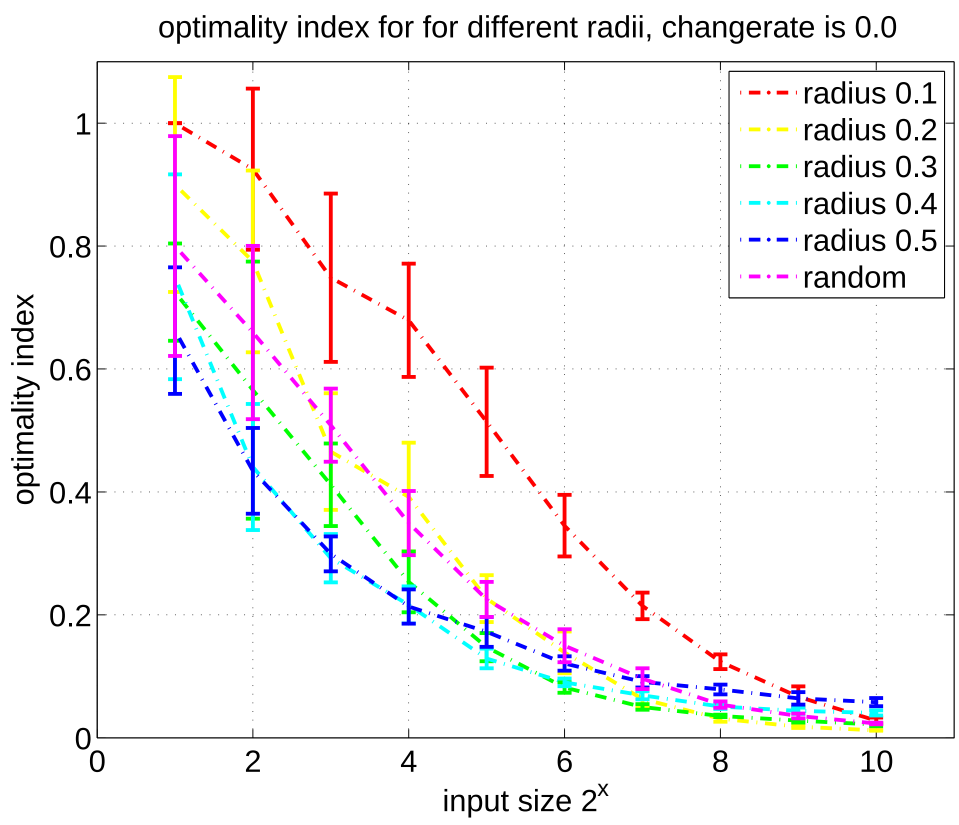
```
    box on
    grid on
    if j~= sizer
        title(sprintf('optimality index for for different n, radius ...
            %1.1f',radius));
    else
        title(sprintf('optimality index for for different n, radius random'));
    end
    %arr = ['r','a','n','d','o','m',' ',' ',' ',' ',' ',' '];
    xlabel('changerate');
    ylabel('optimality index');
    ylim([0,1.1]);
    xlim([-0.1,1.4]);
    legend(num2str(t','n = 2^{%2d}'));
    saveas(handle,sprintf('%s/figure_%d.pdf', dirname, j+6));
end

% surfs
handle = figure(13);
[a,b]=meshgrid(1:10,0.1:0.1:0.5);
hold on;
mm = squeeze(mean(data(:,1:5,3,:,4),4));
view(0,90);
box on
grid on
surf(a,b,mm','EdgeColor','none');
colorbar;
set(gca,'FontSize',16);
xlim([1,10]);
xlabel('input size2^x');
ylabel('radius');
saveas(handle,sprintf('%s/figure_13.pdf', dirname));
end
%generaterandom
handle = figure(14);
set(gca,'FontSize',16);
box on
dd=zeros(10,20,4);
for i=1:10
    n = 2^i;
    for j=1:20
        [a,b] = generateRandom(n);
        [x,y] = makeMatch(a,b);
        dd(i,j,:) = y;
        fprintf('.');
    end
    fprintf('\n');
end
mm = squeeze(mean(dd(:,:,4),2));
st = squeeze(std(dd(:,:,4),0,2));
```

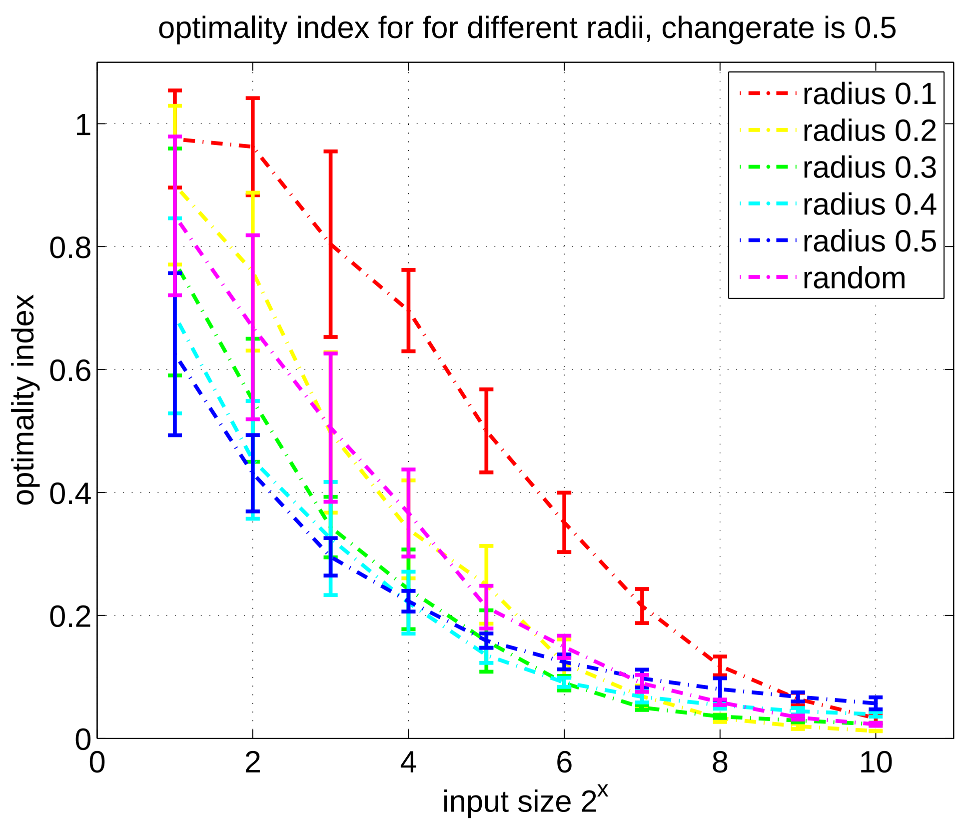
```
errorbar(1:10,mm,st, 'marker', '.','linestyle','-.', 'LineWidth', 2);
%plot(1:10,log2(mm), 'marker', '.','linestyle','-.', 'LineWidth', 2);
p = polyfit(1:10,log2(mm'),1);
% line = polyval(p,1:10);
hold on;
x=1:10;
plot(x,(2^p(2))*(x.^(p(1))));
p
xlabel('input size 2^x');
ylabel('optimality index');
title('optimality index for randomly generated preferences');
%ylim([0,1.1]);
xlim([0,11]);
box on
grid on
%saveas(handle,sprintf('%s/figure_14.pdf', dirname));
end
```

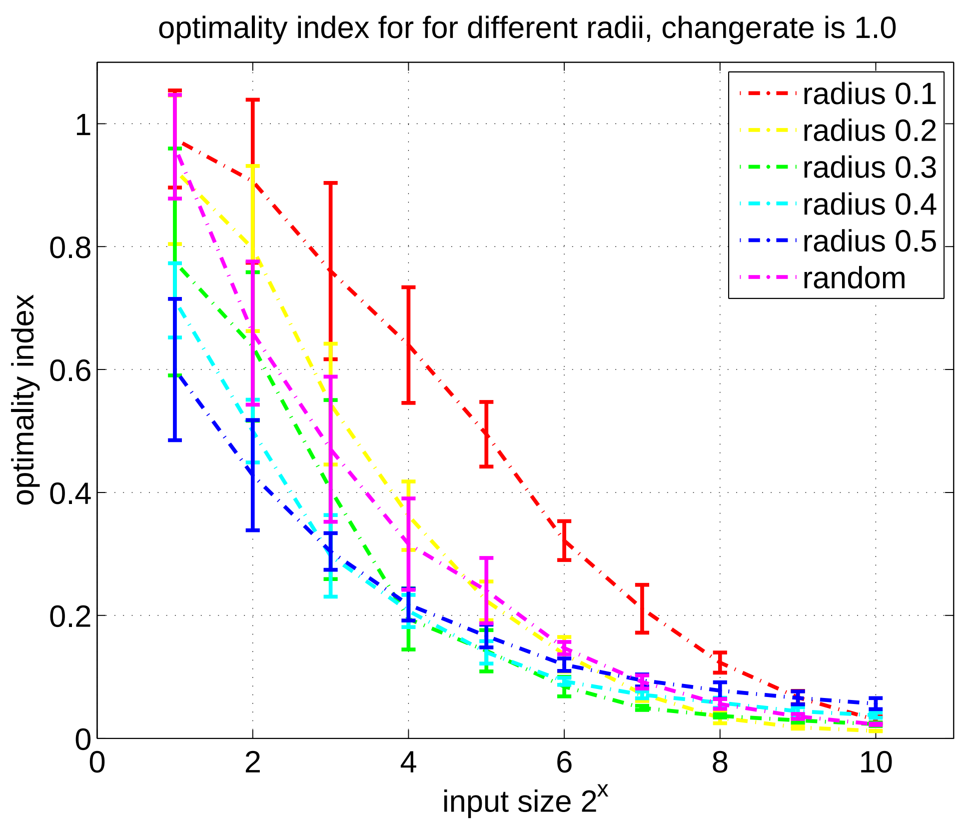
10 Appendix B: Plots

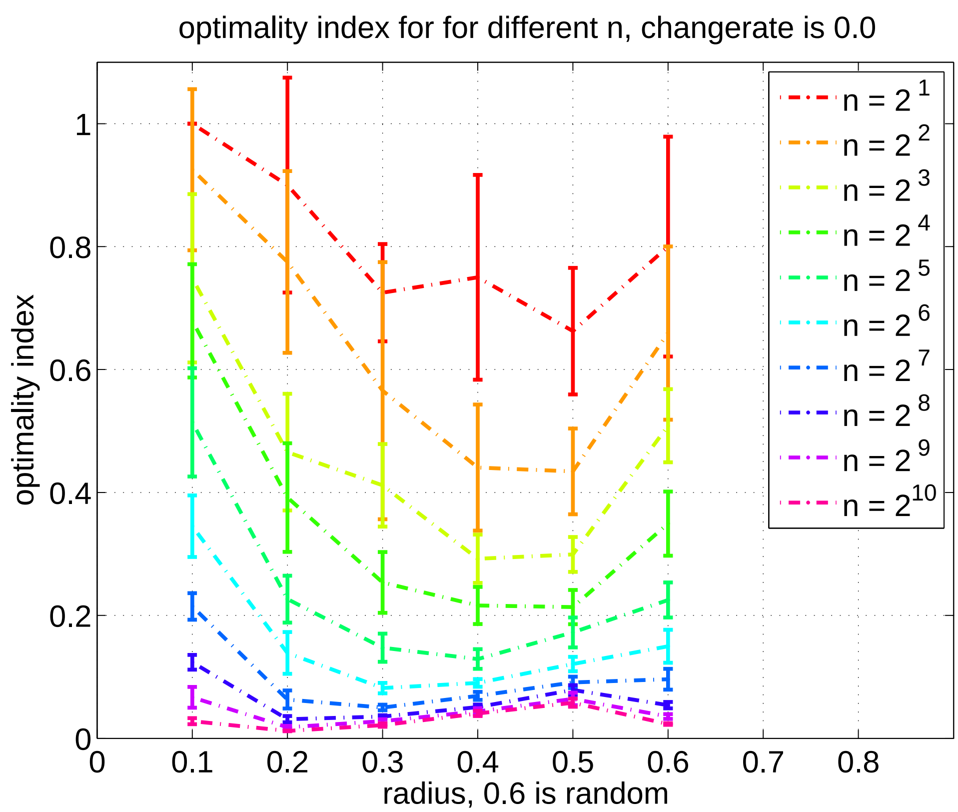
Additional Plots

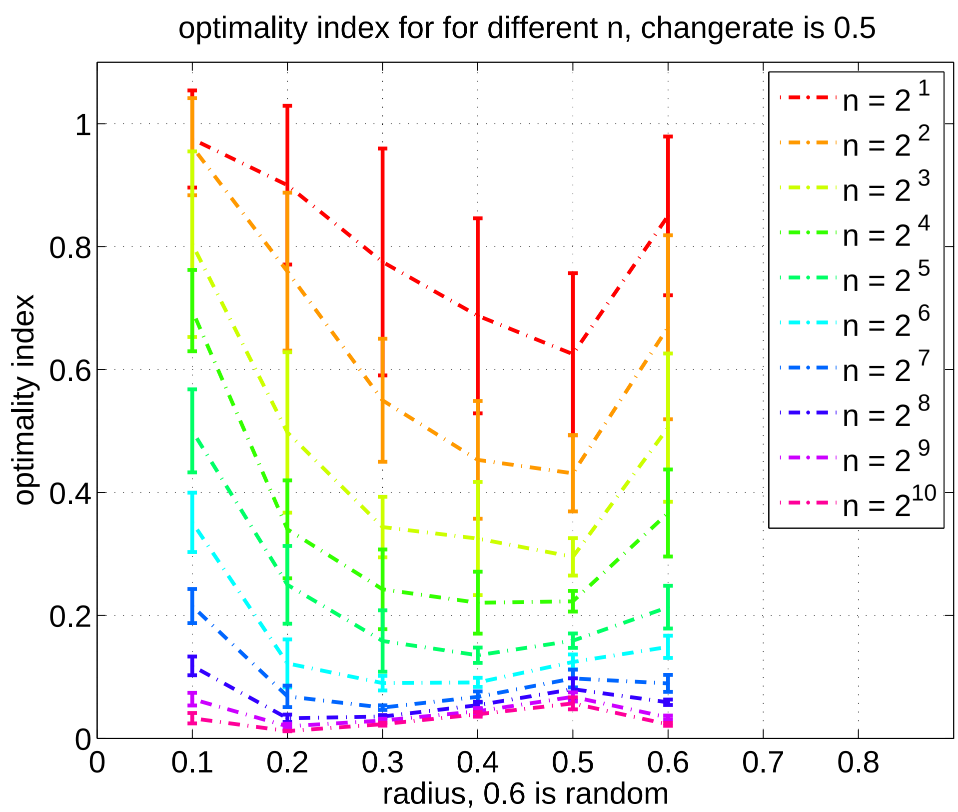
10.1 Optimality Index

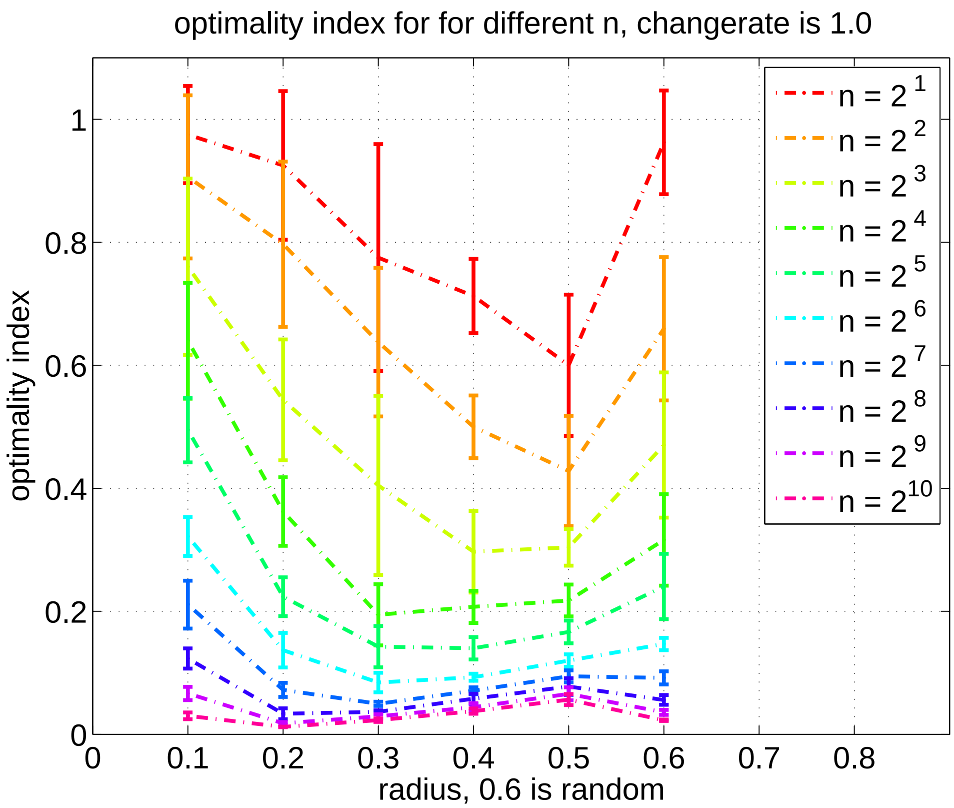



optimality index for for different radii, changerate is 0.5
radius 0.1
radius 0.2
radius 0.3
radius 0.4
radius 0.5
random
optimality index
input size 2^x
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10

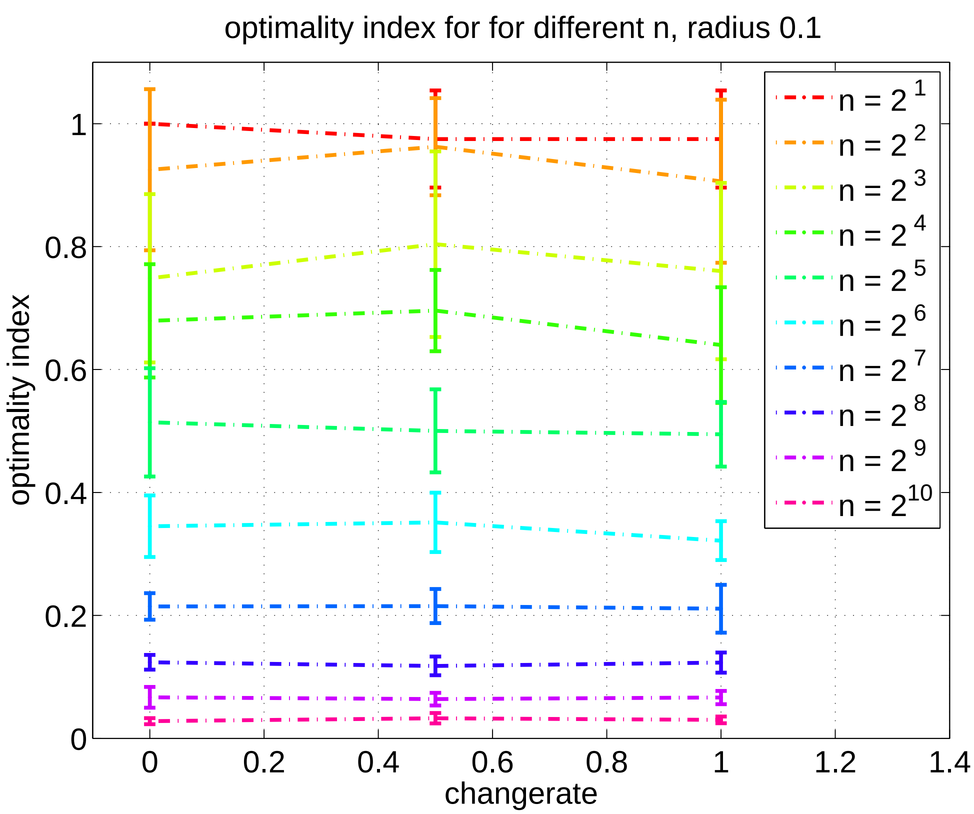

optimality index for for different radii, changerate is 1.0
radius 0.1
radius 0.2
radius 0.3
radius 0.4
radius 0.5
random
optimality index
input size 2^x
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10

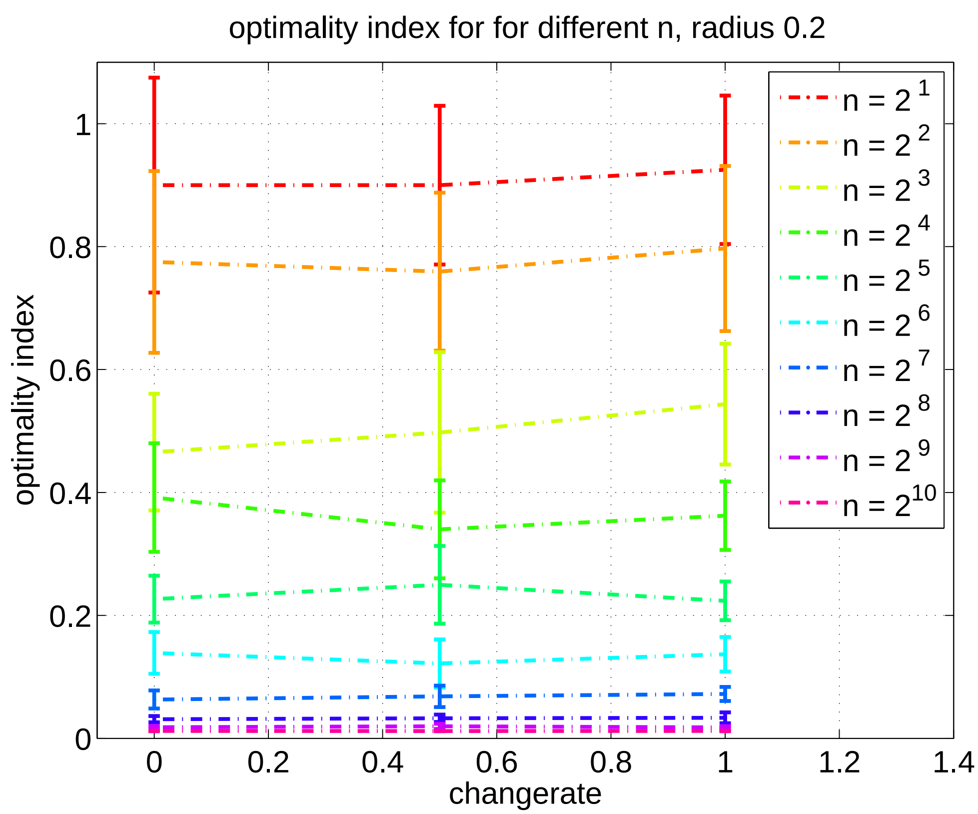

optimality index for for different n, changerate is 0.0
optimality index
radius, 0.6 is random
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10

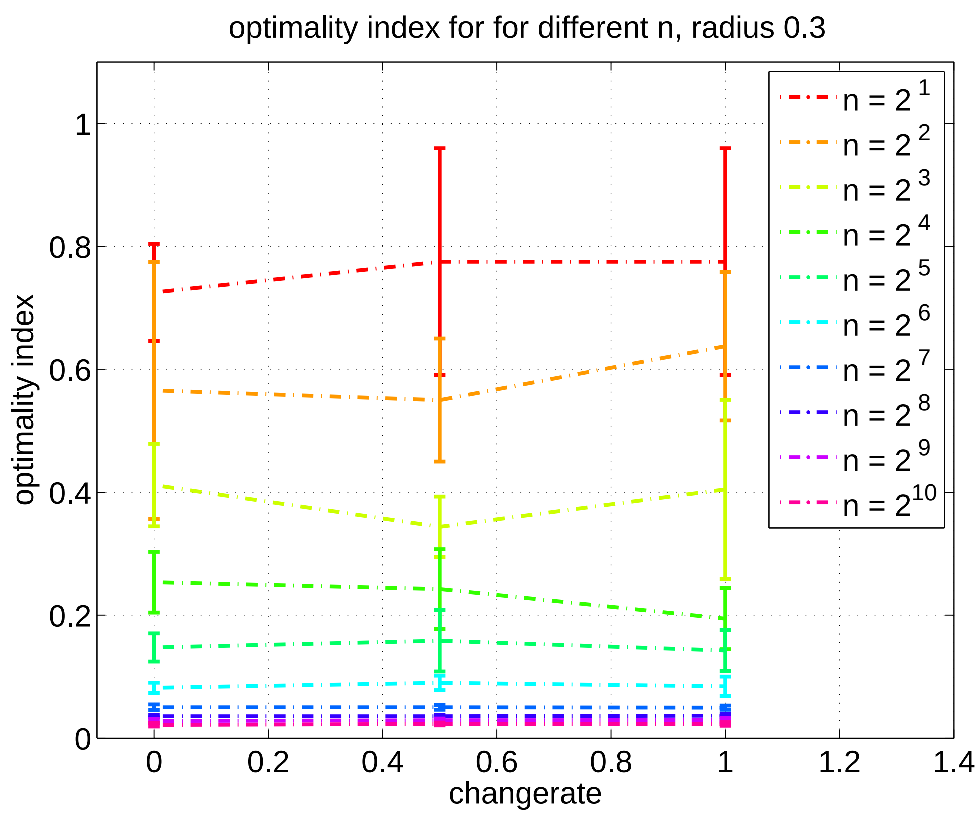

optimality index for for different n, changerate is 0.5
optimality index
radius, 0.6 is random
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10

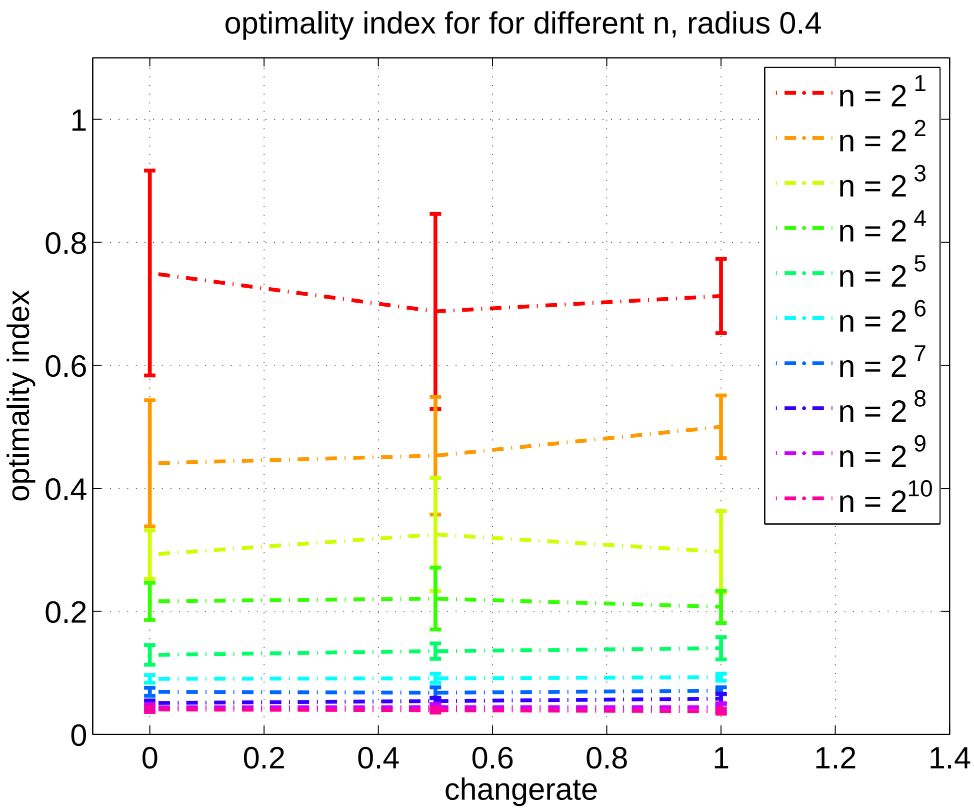


optimality index for for different n, changerate is 1.0
optimality index
radius, 0.6 is random
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10

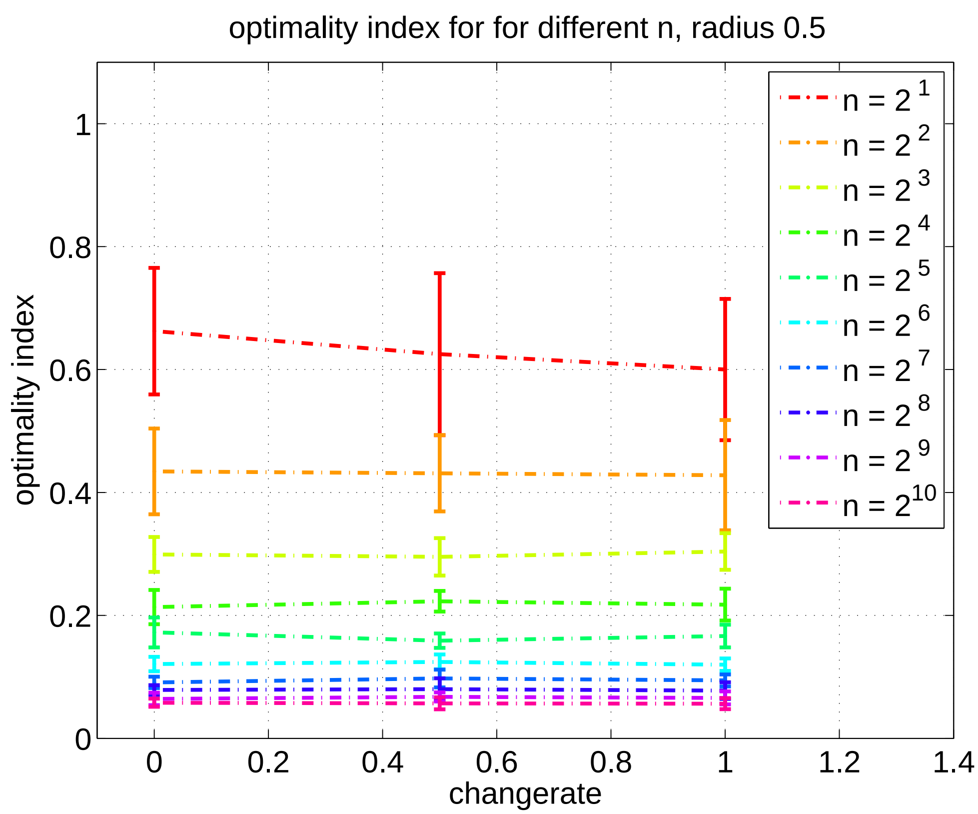

optimality index for for different n, radius 0.1
optimality index
changerate
0
0.2
0.4
0.6
0.8
1
1.2
1.4
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10

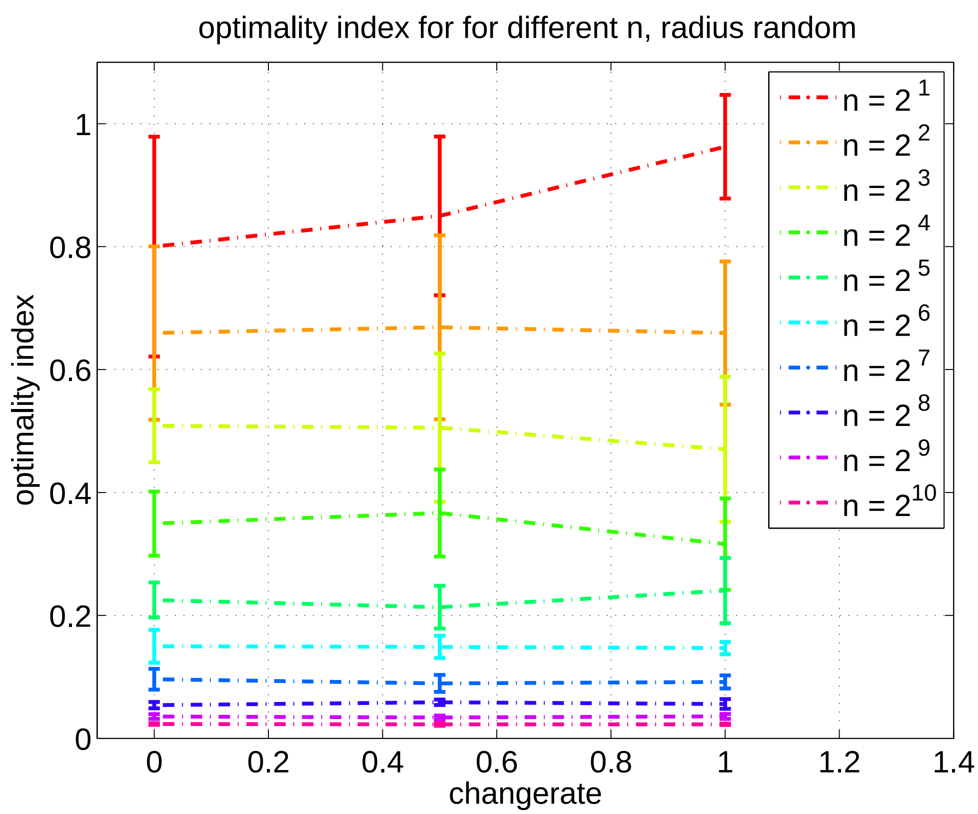

optimality index for for different n, radius 0.2
optimality index
changerate
0
0.2
0.4
0.6
0.8
1
1.2
1.4
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10

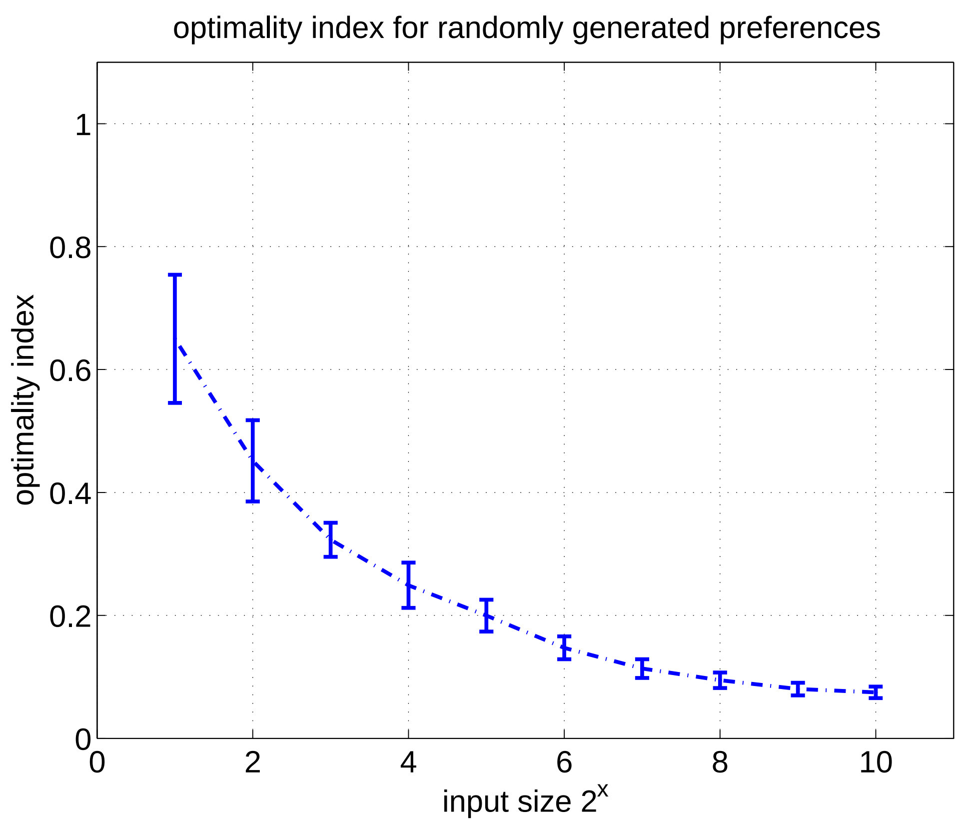

optimality index for for different n, radius 0.3
optimality index
changerate
0
0.2
0.4
0.6
0.8
1
1.2
1.4
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10



optimality index for for different n, radius 0.4
optimality index
changerate
0
0.2
0.4
0.6
0.8
1
1.2
1.4
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10


optimality index for for different n, radius 0.5
optimality index
changerate
0
0.2
0.4
0.6
0.8
1
1.2
1.4
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10


optimality index for for different n, radius random
optimality index
changerate
0
0.2
0.4
0.6
0.8
1
1.2
1.4
n = 2 1
n = 2 2
n = 2 3
n = 2 4
n = 2 5
n = 2 6
n = 2 7
n = 2 8
n = 2 9
n = 2 10


optimality index for randomly generated preferences
optimality index
input size 2^x
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10