Network Design & Algorithm
Final Project

Social Network Design and Algorithm

Dapeng Liu N12261428

Yuanting Song N10731496

May 10th, 2018

Part 1

Topic of Problem

As the increasing of relationships between different people and the developing of online social network systems like facebook, linkedin, more and more people prefer using online social network systems to manage their own social networks. One of the most important functions of social network system is ‘Friend Recommendation’.

Friend Recommendation’s purpose is to recommend new people to be your friends that you may know or you may have familiar background with.

In order to solve this problem, we want to implement a series of models using AMPL to solve this problem. Then if we have other different demands or different ways to determine who to recommend, we could easily modify this model.

Part 2

Basic solution for this problem

1. Instance

①People: Each person is a unit in the system, we will use a node to represent a single person.

②Feature: Features including hobbies, age, gender, study background and so on, and we can also call ‘features’ ‘tags’. Each person could have several features. We also use a node to represent a feature.

③Connection: We have two kinds of connections here. One means two people are already friends, if there is a connection(link) between them. The other one means one person owns a certain feature, if there is a connection(link) between a person and a feature.

2. Correlation

Correlation is a scalar to indicate degree of two people’s similarity, and also the value of correlation is the tendency to recommend these two people to establish connection with each other.

3. Constrains

①Max number of friends: A person could establish connection with others but the number has a upper bound.

②Correlation value of a person: A person’s total correlation value is the sum of the similarity with all others.

③Keep the original topology: The original connections before the algorithm should be kept, new topology should be based on the original topology.

④Undirect connection: If two people connect to each with, then for these two people they all know they are friends.

4. Model

Here we create a new model by using node expressions. Actually, we could use node-link expressions, but it is confusing to set indices for too many links, so here we use node expressions only.

Basic situation model

Indices:

$v = 1, 2, ..., V$	nodes, one node represents one person
$p = 1, 2, ..., P$	features, one feature represents a kind of tag of a person

Constants:

c_v	a set of features that node v connects
$b_{vv'}$	=1 if node v connects node v'; else 0. $v \neq v'$
a_{pv}	=1 if feature p in c_v; else 0.

Variables:

$x_{vv'}$	=1 if node v connects node v' after calculation; else 0. $v \neq v'$
k	total connection a node could have
y_v	correlation value of node v with all other nodes
z_v	second correlation value of node v with all other nodes (some features node v doesn't have but neighbors of node v have)

Objective:

Maximize $F = \sum_v y_v$ (global objective)

Subject to:

$$\sum_{v'} x_{vv'} \le k, v = 1, 2, ..., V$$

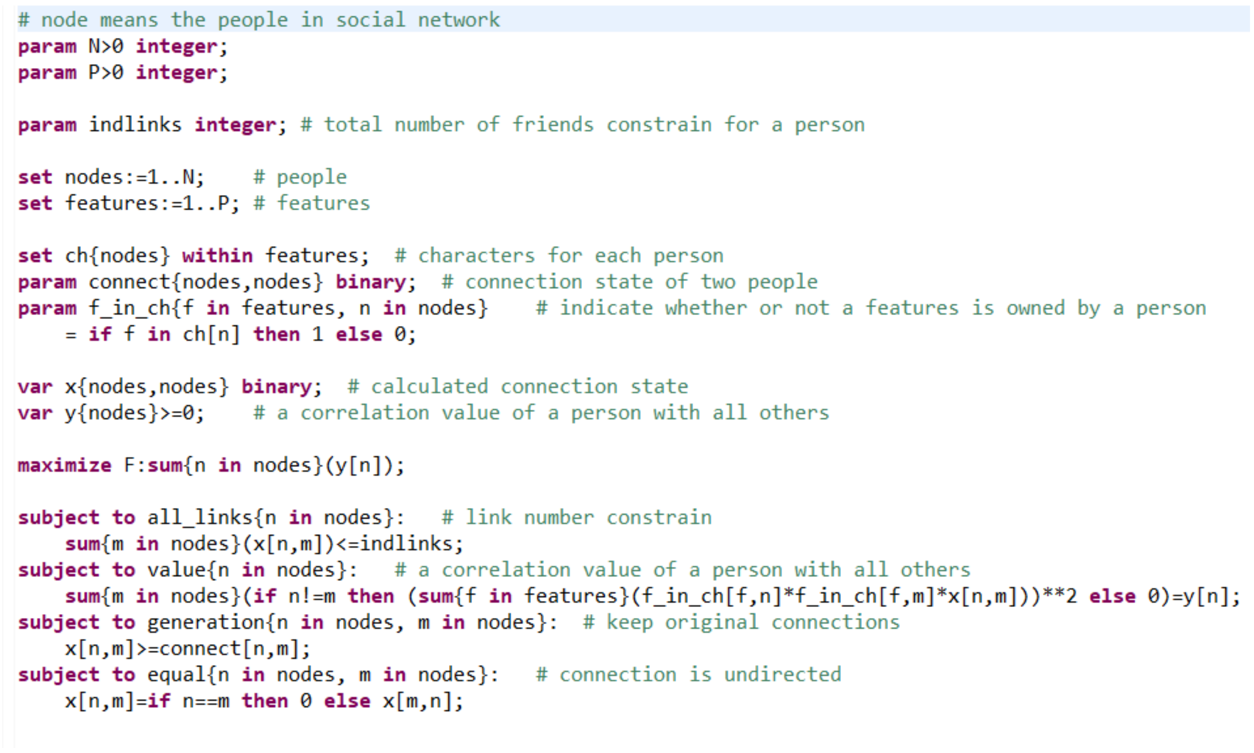
$$\sum_{v' \ne v} (\sum_{p} a_{pv} a_{pv'} x_{vv'})^2 = y_v, v = 1, 2, ..., V$$

$$x_{vv'} \ge b_{vv'}, v = 1, 2, ..., V, v' = 1, 2, ..., V, v \ne v'$$

$$x_{vv'} = x_{v'v}, v = 1, 2, ..., V, v' = 1, 2, ..., V, v \ne v'$$

Here, we use square expression to calculate correlation. Because we want to let two people with more same features to be friends preemptively. (For example 3=2+1, but 9>4+1)

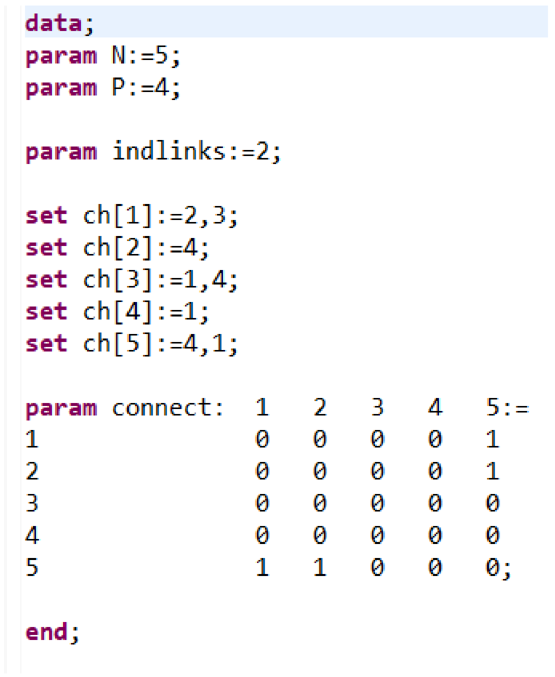
Basic situation AMPL model



```
# node means the people in social network
param N>0 integer;
param P>0 integer;

param indlinks integer; # total number of friends constrain for a person

set nodes:=1..N;     # people
set features:=1..P; # features

set ch{nodes} within features;  # characters for each person
param connect{nodes,nodes} binary;  # connection state of two people
param f_in_ch{f in features, n in nodes}    # indicate whether or not a features is owned by a person
    = if f in ch[n] then 1 else 0;

var x{nodes,nodes} binary;  # calculated connection state
var y{nodes}>=0;     # a correlation value of a person with all others

maximize F:sum{n in nodes}(y[n]);

subject to all_links{n in nodes}:   # link number constrain
    sum{m in nodes}(x[n,m])<=indlinks;
subject to value{n in nodes}:   # a correlation value of a person with all others
    sum{m in nodes}(if n!=m then (sum{f in features}(f_in_ch[f,n]*f_in_ch[f,m]*x[n,m]))**2 else 0)=y[n];
subject to generation{n in nodes, m in nodes}:  # keep original connections
    x[n,m]>=connect[n,m];
subject to equal{n in nodes, m in nodes}:   # connection is undirected
    x[n,m]=if n==m then 0 else x[m,n];
```

Basic situation example

1. Input data:



```
data;
param N:=5;
param P:=4;

param indlinks:=2;

set ch[1]:=2,3;
set ch[2]:=4;
set ch[3]:=1,4;
set ch[4]:=1;
set ch[5]:=4,1;

param connect:  1   2   3   4   5:=
1               0   0   0   0   1
2               0   0   0   0   1
3               0   0   0   0   0
4               0   0   0   0   0
5               1   1   0   0   0;

end;
```

5 people, 4 features, each person could have no more than 2 friends

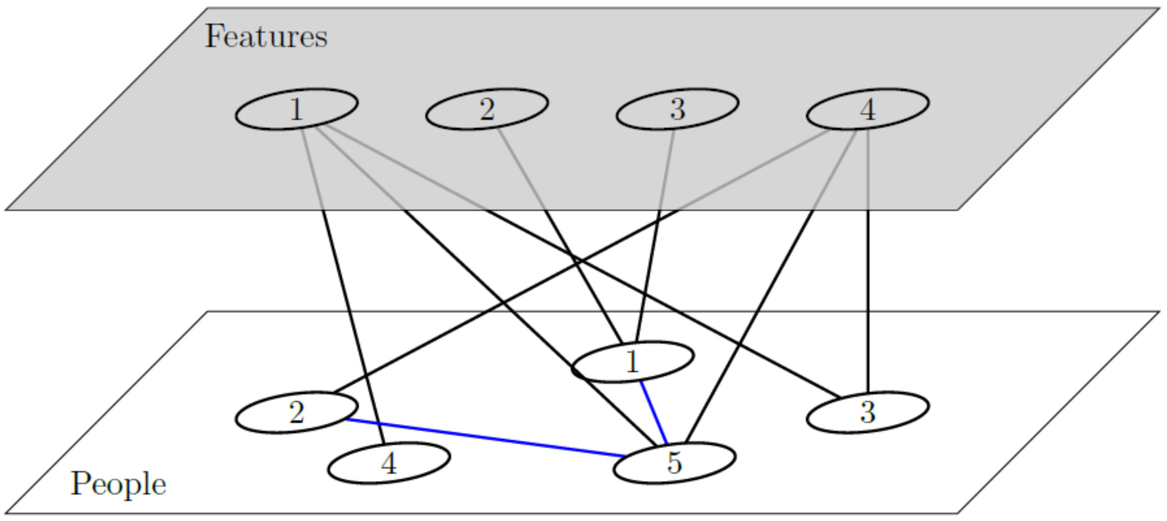
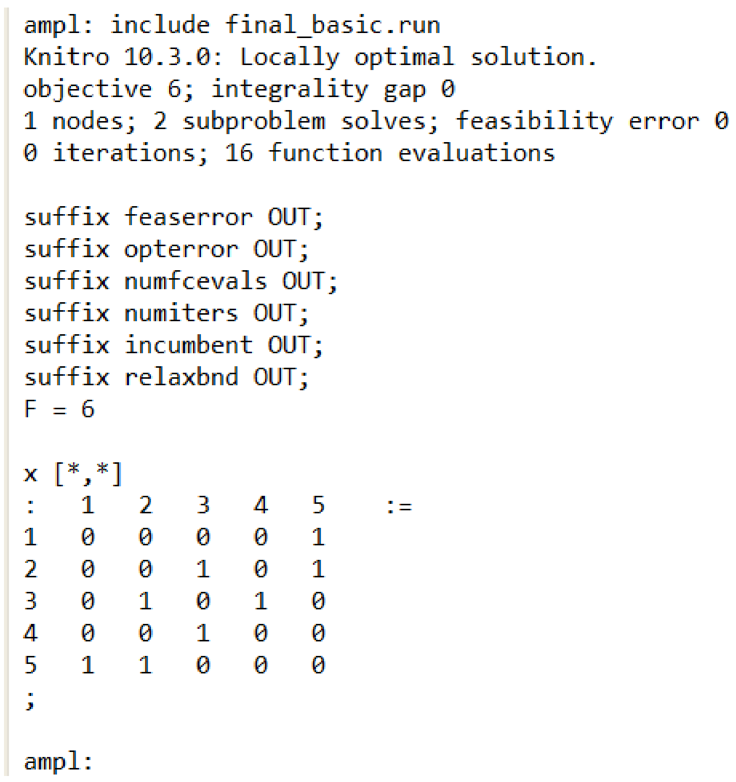


Fig.1 Input state

2. Output result:

p.s: In order to run these models, we need a full version of AMPL and we need solver 'Knitro'.



```
ampl: include final_basic.run
Knitro 10.3.0: Locally optimal solution.
objective 6; integrality gap 0
1 nodes; 2 subproblem solves; feasibility error 0
0 iterations; 16 function evaluations

suffix feaserror OUT;
suffix opterror OUT;
suffix numfcevals OUT;
suffix numiters OUT;
suffix incumbent OUT;
suffix relaxbnd OUT;
F = 6

x [*,*]
:   1   2   3   4   5    :=
1   0   0   0   0   1
2   0   0   1   0   1
3   0   1   0   1   0
4   0   0   1   0   0
5   1   1   0   0   0
;

ampl:
```

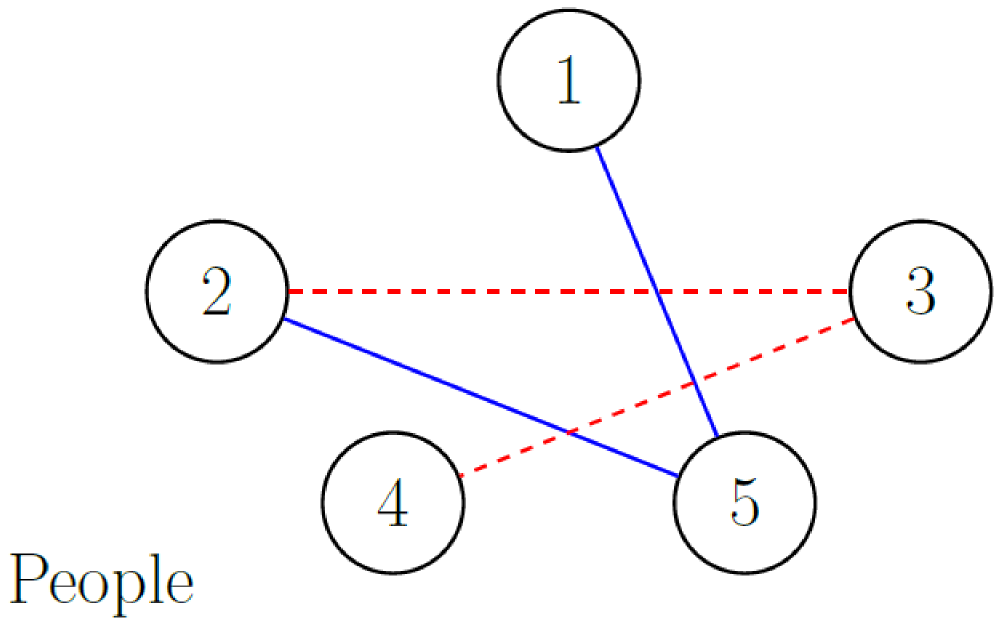


Fig.2 Output result of basic model

According to the output result, it means the algorithm recommend 2-3 and 3-4 to be friends(blue lines are original connections, red lines are recommendation lines).

Part 3

Friend impact model based on basic model

In basic model, we only consider the simplest condition that we recommend two people to become friends just based on these two people's similarity. Obviously, it is not enough.

Now consider another condition which is more complex: A and B are friends, A likes soccer while B likes basketball. Then we have C and D, both of them are not A's friends and also they are not interested in soccer, but C likes baseball and D likes basketball. Now if I want to recommend new friend C or D to A, who should be choose?

In this condition, we should recommend D to A, because although A doesn't like basketball, A's friend B likes basketball. And we should call this 'Friend Impact'.

So now let's solve this problem.

Friend impact model

Indices:

$v = 1, 2, ..., V$	nodes, one node represents one person
$p = 1, 2, ..., P$	features, one feature represents a kind of tag of a person

Constants:

c_v	a set of features that node v connects
$b_{vv'}$	=1 if node v connects node v'; else 0. $v \neq v'$
a_{pv}	=1 if feature p in c_v; else 0.

Variables:

$x_{vv'}$	=1 if node v connects node v' after calculation; else 0. $v \neq v'$
$\delta_{vv'p}$	$= b_{vv'}$ if p doesn't belong to $c_{v'}$ and belongs to c_v; else 0. $v \neq v'$

k total connection a node could have

y_v correlation value of node v with all other nodes

z_v second correlation value of node v with all other nodes (some features node v doesn't have but neighbors of node v have)

Objective:

$$\text{Minimize } F = \sum_v (y_v + 0.1 * z_v)$$

Subject to:

$$\sum_{v'} x_{vv'} \le k, v = 1, 2, \ldots, V$$

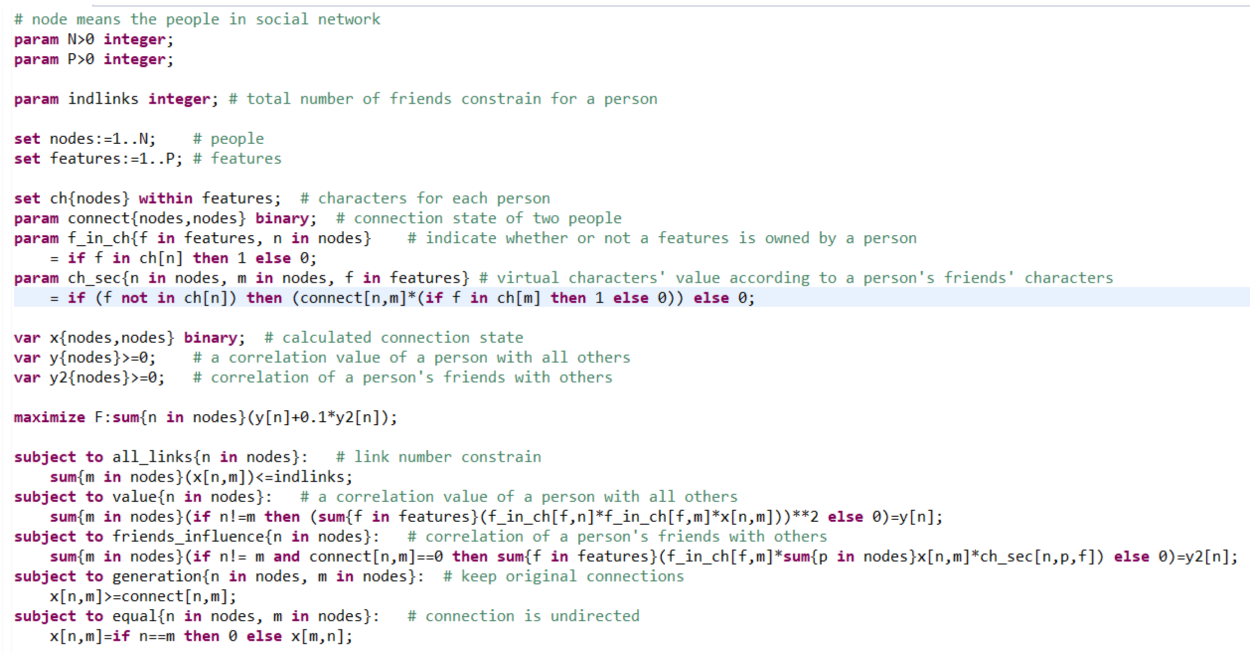
$$\sum_{v' \ne v} (\sum_p a_{pv} a_{pv'} x_{vv'})^2 = y_v, v = 1, 2, \ldots, V$$

$$\sum_{v' \ne v} (1 - b_{vv'})(\sum_p (a_{pv'} \sum_{v''} x_{vv'} \delta_{vv''p})) = z_v, v = 1, 2, \ldots, V$$

$$x_{vv'} \ge b_{vv'}, v = 1, 2, \ldots, V, v' = 1, 2, \ldots, V, v \ne v'$$

$$x_{vv'} = x_{v'v}, v = 1, 2, \ldots, V, v' = 1, 2, \ldots, V, v \ne v'$$

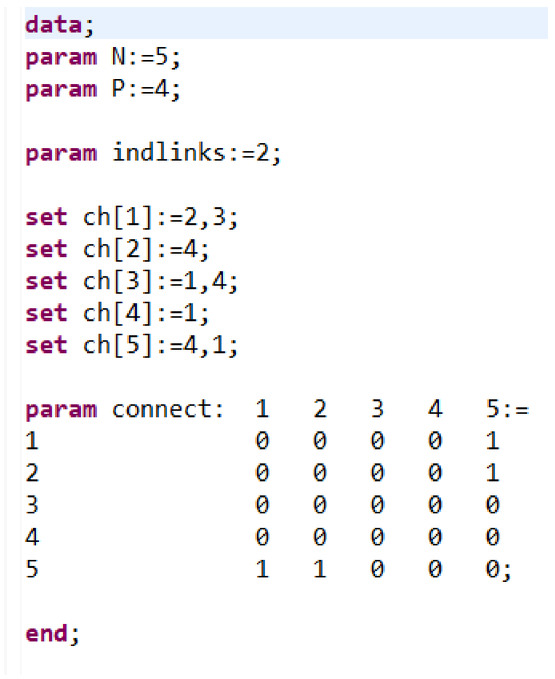
Friend impact AMPL model



```
# node means the people in social network
param N>0 integer;
param P>0 integer;

param indlinks integer; # total number of friends constrain for a person

set nodes:=1..N;    # people
set features:=1..P; # features

set ch{nodes} within features;  # characters for each person
param connect{nodes,nodes} binary;  # connection state of two people
param f_in_ch{f in features, n in nodes}    # indicate whether or not a features is owned by a person
    = if f in ch[n] then 1 else 0;
param ch_sec{n in nodes, m in nodes, f in features} # virtual characters' value according to a person's friends' characters
    = if (f not in ch[n]) then (connect[n,m]*(if f in ch[m] then 1 else 0)) else 0;

var x{nodes,nodes} binary;  # calculated connection state
var y{nodes}>=0;    # a correlation value of a person with all others
var y2{nodes}>=0;   # correlation of a person's friends with others

maximize F:sum{n in nodes}(y[n]+0.1*y2[n]);

subject to all_links{n in nodes}:   # link number constrain
    sum{m in nodes}(x[n,m])<=indlinks;
subject to value{n in nodes}:   # a correlation value of a person with all others
    sum{m in nodes}(if n!=m then (sum{f in features}(f_in_ch[f,n]*f_in_ch[f,m]*x[n,m]))**2 else 0)=y[n];
subject to friends_influence{n in nodes}:   # correlation of a person's friends with others
    sum{m in nodes}(if n!= m and connect[n,m]==0 then sum{f in features}(f_in_ch[f,m]*sum{p in nodes}x[n,m]*ch_sec[n,p,f]) else 0)=y2[n];
subject to generation{n in nodes, m in nodes}:  # keep original connections
    x[n,m]>=connect[n,m];
subject to equal{n in nodes, m in nodes}:   # connection is undirected
    x[n,m]=if n==m then 0 else x[m,n];
```

Friend impact example

1. Input data:



```
data;
param N:=5;
param P:=4;

param indlinks:=2;

set ch[1]:=2,3;
set ch[2]:=4;
set ch[3]:=1,4;
set ch[4]:=1;
set ch[5]:=4,1;

param connect:  1   2   3   4   5:=
1               0   0   0   0   1
2               0   0   0   0   1
3               0   0   0   0   0
4               0   0   0   0   0
5               1   1   0   0   0;

end;
```

5 people, 4 features, each person could have no more than 2 friends (Same as basic example)

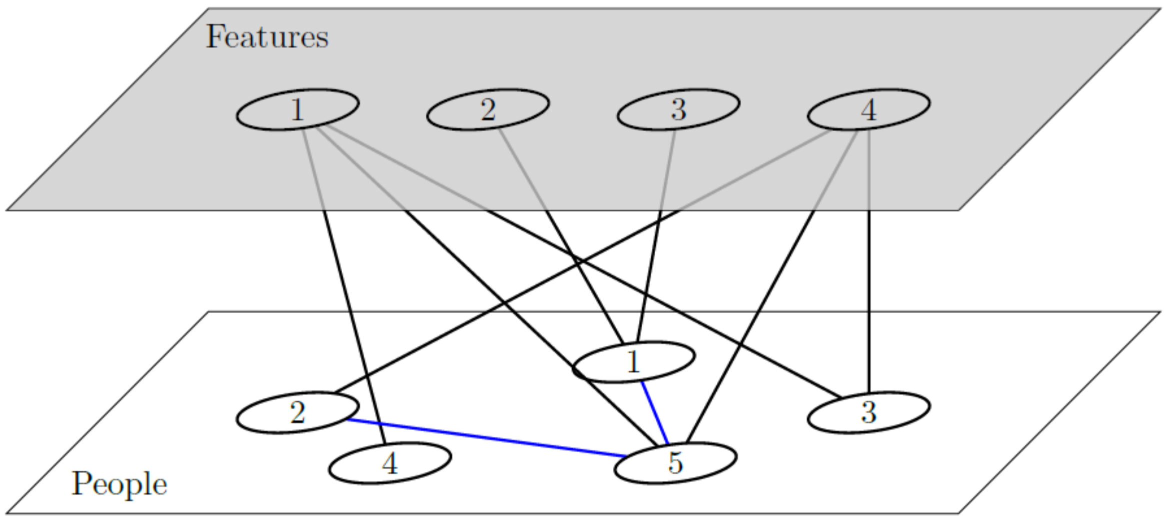
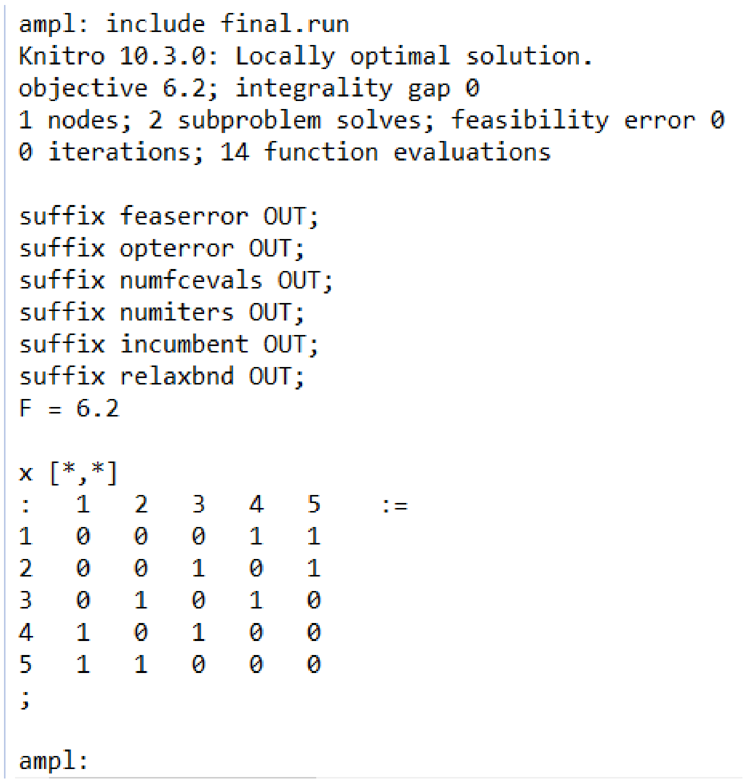


Fig.3 Input state

2. Output result:



```
ampl: include final.run
Knitro 10.3.0: Locally optimal solution.
objective 6.2; integrality gap 0
1 nodes; 2 subproblem solves; feasibility error 0
0 iterations; 14 function evaluations

suffix feaserror OUT;
suffix opterror OUT;
suffix numfcevals OUT;
suffix numiters OUT;
suffix incumbent OUT;
suffix relaxbnd OUT;
F = 6.2

x [*,*]
:   1   2   3   4   5    :=
1   0   0   0   1   1
2   0   0   1   0   1
3   0   1   0   1   0
4   1   0   1   0   0
5   1   1   0   0   0
;

ampl:
```

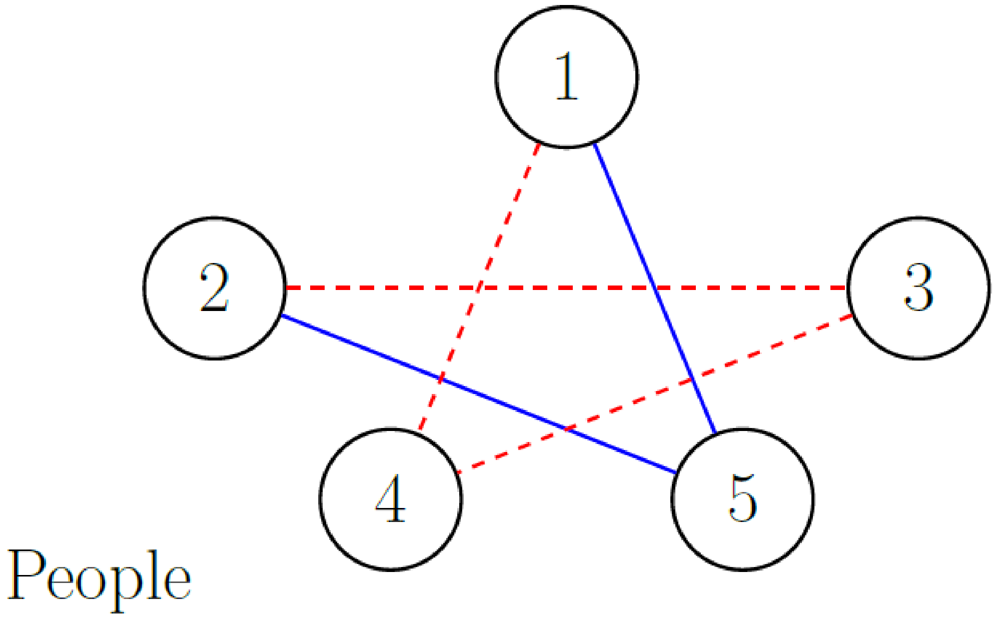


Fig.4 Output result of friend impact model

According to the output result, it means the algorithm recommend 1-4, 2-3, and 3-4 to be friends.

Part 4

Adding constrain model based on Friend impact model

Now based on Friend impact model, we should consider why each person could only have limited number of friends. What if not?

Actually, in real world, we can see no matter how many friends a person already had, the system should also recommend some new friends to him/her, you will never see an empty friend recommendation page in your linkedin or facebook.

To solve this problem, we should change our friend impact model further. In fact, it's quite easy, just need to change 'total number of friends constrain' to 'adding number of friends constrain'.

Adding constrain model

Indices:

$v = 1, 2, ..., V$	nodes, one node represents one person
$p = 1, 2, ..., P$	features, one feature represents a kind of tag of a person

Constants:

c_v	a set of features that node v connects
$b_{vv'}$	=1 if node v connects node v'; else 0. $v \neq v'$
a_{pv}	=1 if feature p in c_v; else 0.

Variables:

$x_{vv'}$	=1 if node v connects node v' after calculation; else 0. $v \neq v'$
$\delta_{vv'p}$	$= b_{vv'}$ if p doesn't belong to $c_{v'}$ and belongs to c_v; else 0. $v \neq v'$
k	upper bound of adding number of friends in one step for a person

y_v correlation value of node v with all other nodes

z_v second correlation value of node v with all other nodes (some features node v doesn't have but neighbors of node v have)

Objective:

Minimize $F = \sum_{v} (y_v + 0.1 * z_v)$

Subject to:

$$\sum_{v'} (x_{vv'} - b_{vv'}) \le k, v = 1, 2, ..., V$$

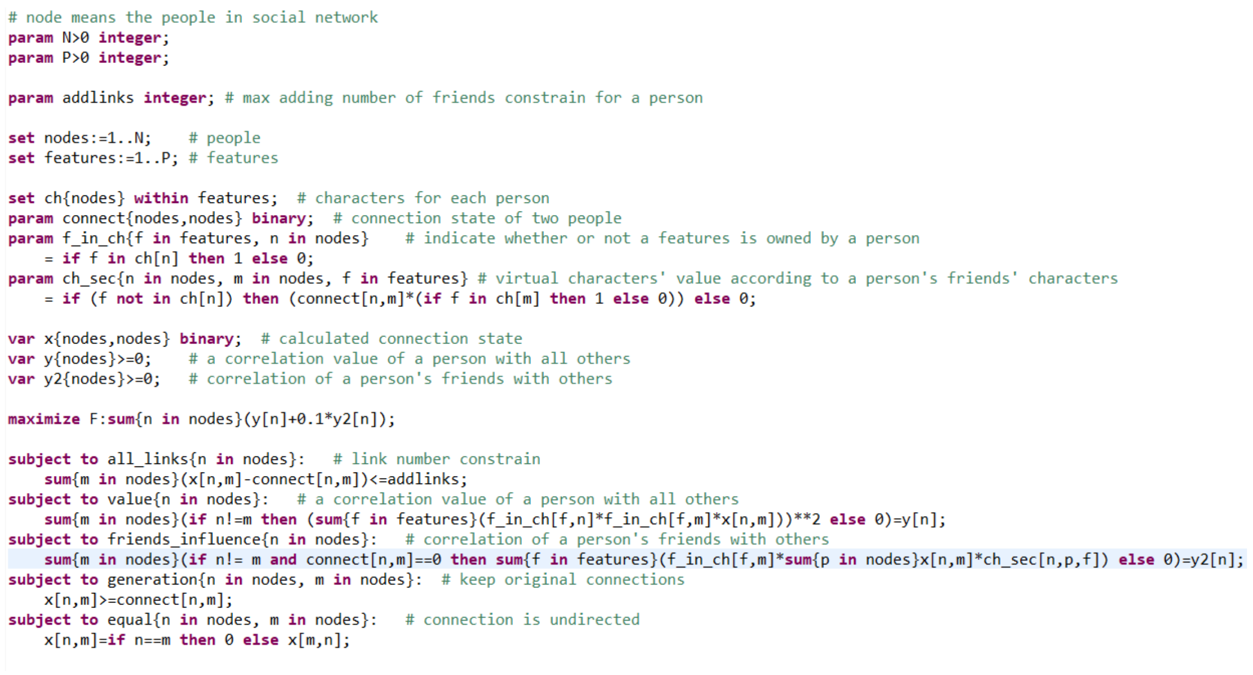
$$\sum_{v' \ne v} (\sum_{p} a_{pv} a_{pv'} x_{vv'})^2 = y_v, v = 1, 2, ..., V$$

$$\sum_{v' \ne v} (1 - b_{vv'})(\sum_{p} (a_{pv'} \sum_{v''} x_{vv'} \delta_{vv''p})) = z_v, v = 1, 2, ..., V$$

$$x_{vv'} \ge b_{vv'}, v = 1, 2, ..., V, v' = 1, 2, ..., V, v \ne v'$$

$$x_{vv'} = x_{v'v}, v = 1, 2, ..., V, v' = 1, 2, ..., V, v \ne v'$$

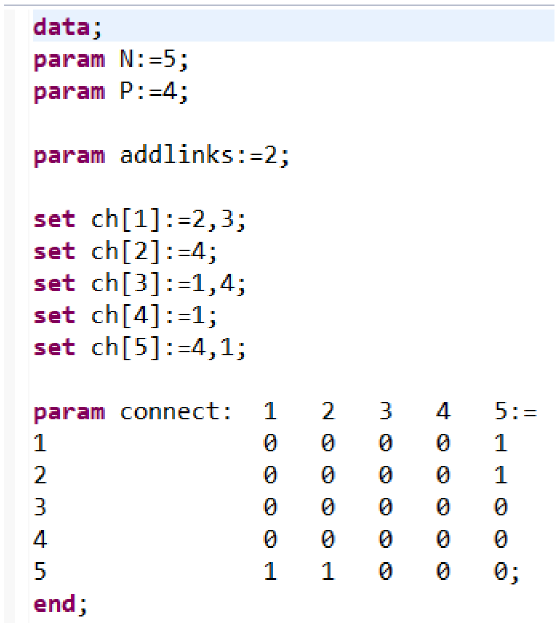
Adding constrain AMPL model



```
# node means the people in social network
param N>0 integer;
param P>0 integer;

param addlinks integer; # max adding number of friends constrain for a person

set nodes:=1..N;    # people
set features:=1..P; # features

set ch{nodes} within features;  # characters for each person
param connect{nodes,nodes} binary;  # connection state of two people
param f_in_ch{f in features, n in nodes}    # indicate whether or not a features is owned by a person
    = if f in ch[n] then 1 else 0;
param ch_sec{n in nodes, m in nodes, f in features} # virtual characters' value according to a person's friends' characters
    = if (f not in ch[n]) then (connect[n,m]*(if f in ch[m] then 1 else 0)) else 0;

var x{nodes,nodes} binary;  # calculated connection state
var y{nodes}>=0;    # a correlation value of a person with all others
var y2{nodes}>=0;   # correlation of a person's friends with others

maximize F:sum{n in nodes}(y[n]+0.1*y2[n]);

subject to all_links{n in nodes}:   # link number constrain
    sum{m in nodes}(x[n,m]-connect[n,m])<=addlinks;
subject to value{n in nodes}:   # a correlation value of a person with all others
    sum{m in nodes}(if n!=m then (sum{f in features}(f_in_ch[f,n]*f_in_ch[f,m]*x[n,m]))**2 else 0)=y[n];
subject to friends_influence{n in nodes}:   # correlation of a person's friends with others
    sum{m in nodes}(if n!= m and connect[n,m]==0 then sum{f in features}(f_in_ch[f,m]*sum{p in nodes}x[n,m]*ch_sec[n,p,f]) else 0)=y2[n];
subject to generation{n in nodes, m in nodes}:  # keep original connections
    x[n,m]>=connect[n,m];
subject to equal{n in nodes, m in nodes}:   # connection is undirected
    x[n,m]=if n==m then 0 else x[m,n];
```

Adding constrain example

1. Input data:



```
data;
param N:=5;
param P:=4;

param addlinks:=2;

set ch[1]:=2,3;
set ch[2]:=4;
set ch[3]:=1,4;
set ch[4]:=1;
set ch[5]:=4,1;

param connect:  1   2   3   4   5:=
1               0   0   0   0   1
2               0   0   0   0   1
3               0   0   0   0   0
4               0   0   0   0   0
5               1   1   0   0   0;
end;
```

5 people, 4 features, each person could add no more than 2 friends once

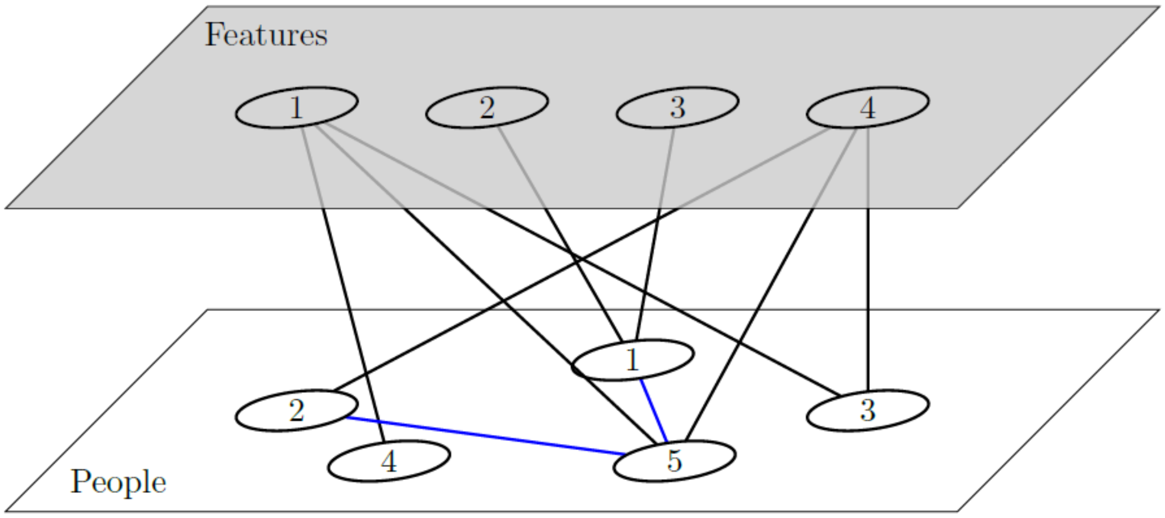
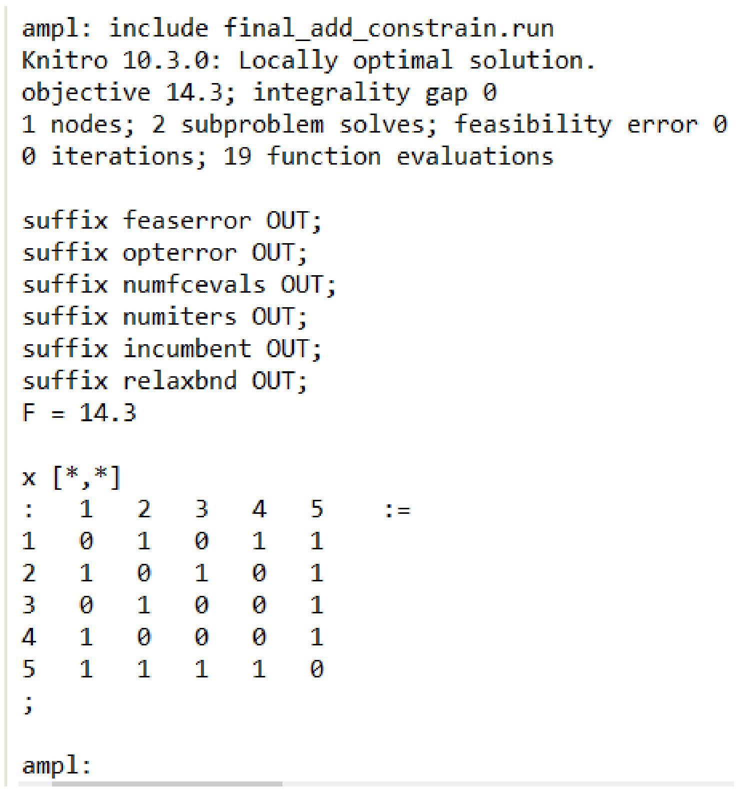


Fig.5 Input state

2. Output result:



```
ampl: include final_add_constrain.run
Knitro 10.3.0: Locally optimal solution.
objective 14.3; integrality gap 0
1 nodes; 2 subproblem solves; feasibility error 0
0 iterations; 19 function evaluations

suffix feaserror OUT;
suffix opterror OUT;
suffix numfcevals OUT;
suffix numiters OUT;
suffix incumbent OUT;
suffix relaxbnd OUT;
F = 14.3

x [*,*]
:   1   2   3   4   5    :=
1   0   1   0   1   1
2   1   0   1   0   1
3   0   1   0   0   1
4   1   0   0   0   1
5   1   1   1   1   0
;

ampl:
```

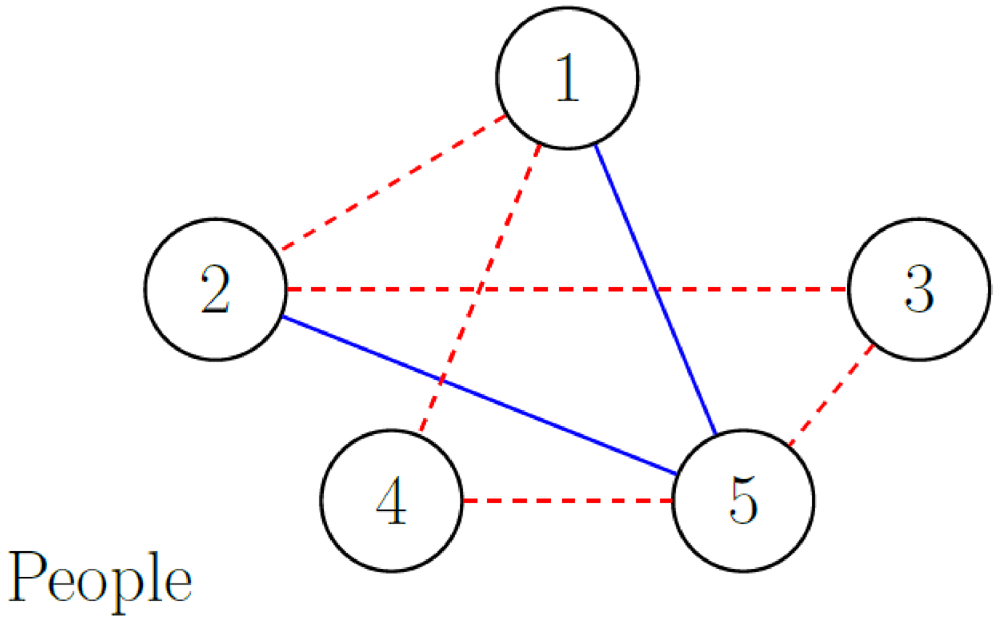


Fig.6 Output result of adding constrain model

According to the output result, it means the algorithm recommend 1-2, 1-4, 2-3, 3-5 and 4-5 to be friends.

So, if now we watch the pages of friend recommendation on website:

For 1: You may know 2 and 4;

For 2: You may know 1 and 3;

For 3: You may know 2 and 5;

For 4: You may know 1 and 5;

For 5: You may know 3 and 4;

Now our system is really close to the real friend recommendation system!

Part 5

Conclusion and more ideas

Our models solved some basic problems of friend recommendation, once everyone fill the form of his/her features he/she will easily get the friend recommendation by using these models.

Now the models' complexity is not very large, and also the models could append more complex functions and specific algorithms to let the models get more accurate results.

For example, a person's friends' friends could also have influence to him/her, we could introduce a series of regression coefficients to distribute different weight to these influence; and also if two people are not friends but they have shared friends, then these two people should have more chance to be friends.