Distributed Signal Processing Fire Detection Project Report

Yuhan Xu(4582489), Jinzi Qi(4630939)

Delft University of Technology

April 18, 2017

I. Introduction

Nowadays, due to the explosion in size and complexity of dataset, solving problems with very large size database becomes a hot research area.[1] Distributed processing technology is able to deal with cases in high complexity and large dataset and can increase the reliability and performance of the system. As a result, distributed technology is widely used in many fields, especially in signal processing, such as network analysis, weather forecast, telecommunication and so on.

Since forest fire can lead to enormous damages, detecting it timely when it is still controllable is quite important for loss control. In this report, distributed signal processing is applied in fire detection. The region to be detected is a $30m \times 60m$ plane and the sensors are installed on the ceiling to monitor the environmental temperature. The sensors communicate via wireless data transmission and has a transmission radius $3m$. To make the problem simple, the temperature in the plant is assumed to follow a Gaussian distribution, i.e., $X\ N(20, \sigma^2)$ in normal and $X\ N(20, \sigma^2)$ in the fire region. In addition, there is no central unit to collect and process the data and all computations and transmissions are carried over the sensors.

To solve the problem, firstly sensor network in hexagonally distribution is designed optimally to cover all area in specified plant. The temperature detected by the sensor work is averaged by two algorithms, randomized gossip and greedy gossip with eavesdropping. The results obtained by the two algorithms are compared while the convergence speed and possible failures conditions are also analyzed. Then to detect the region on fire, MAP detector is applied to find the optimal determination of a state in a sensor area. The influence of variances in temperature distribution is also considered.

The report is constructed in the following order: Chapter 2 will introduce the sensor network topology designed in the report; Chapter 3 will provide the models and algorithms used in the report; Chapter 4 presents the experimental results and analysis to them; and Chapter 5 gives the conclusions.

II. Sensor Network Topology

This section mainly discusses the sensor network topology used to allocate the sensors. As mentioned in section 1, the sensors will be placed in vertexes of each hexagon and the target area is covered by coherent hexagons which is shown in Figure 1.

The side length of the hexagon is $3m$ which is identical to the transmission radius of the sensors. The black point in the vertex of hexagon donotes the sensor. The red number in the figure is the serial number of each sensor and there are 168 sensors in total.

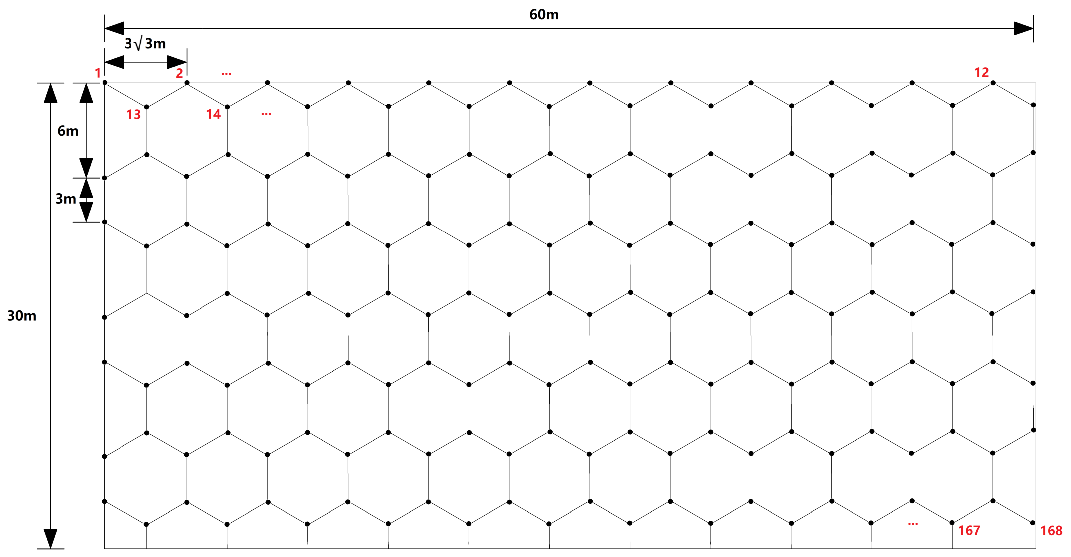


Figure 1: *The sensor network topology.*

By using this network topology, every corner of this map can be covered and the temperatures of the whole region can be detected by sensors. One of the advantage of the hexagonal distribution is that it can reduce the number of sensors effectively compared to the rectangular distribution with at least $21 \times 11 = 231$ sensors. Also, as each sensor has 3 neighbors at most, the averaging efficiency increases compared to rectangular topology whose sensor has 4 neighbors at most. Although the location information in hexagon topology can be kind of complex, the utilization of such distribution has more benefits.

III. Averaging and Detection Method

In this section, two different models of averaging which are randomized gossip and greedy gossip is introduced. And the fire detection method based on MAP detector is proposed as well.

i. Averaging Algorithms

The average among sensors is carried out by two kinds of gossip algorithms respectively.

i.1 Randomized Gossip

Randomized gossip algorithm is an asynchronous distributed averaging algorithm that does not need any global knowledge of the network. In one time slot, each sensor is only allowed to communicate with one of its neighbors in a certain probability and exchange their values while refresh their values with the mean of two original values. In order to simplify the calculation in the report, the probability of each neighbor to be chosen is equal. Then after several iteration, the value in each sensor will be close to the average value of the entire network. With number of iterations large enough, the goal to compute the average temperature of the plant is possible to achieve.

The main step of the algorithm can be indicated as following, where n is the number of sensors,

Algorithm 1 Randomized Gossip

```
Input: original sensor values, neighbor information of sensors, maximal iteration number $k_{max}$
Output: averaged sensor value, error
1: In $k$th time slot, select a sensor $i$ at random (uniform distribution, probability $1/n$) and let it
   contact some neighbouring node $j$ with equal probability
2: At this time, both nodes set their values equal to the average of their current values
3: Stop the algorithm if $k >= k_{max}$
```

i.2 Greedy Gossip with Eavesdropping

Greedy gossip algorithm with eavesdropping is similar to randomized gossip algorithm. The difference is that the greedy gossip implements a greedy neighbor selection procedure. It only exchanges the value with the neighbor whose value differs mostly, and then their values are updated with the mean of their two. The specific step of this algorithm is provided as following.

Algorithm 2 Greedy Gossip with Eavesdropping

```
Input: original sensor values, neighbor information of sensors, maximal iteration number $k_{max}$
Output: averaged sensor value, error
1: In $k$th time slot, select a sensor $i$ at random (uniform distribution, probability $1/n$) and let it
   contact the neighbor who has the most different value from its own
2: At this time, both nodes set their values equal to the average of their current values
3: Stop the algorithm if $k >= k_{max}$
```

ii. Fire Detection Method

To detect fire, the statement of the sensor area need to be determined first. Suppose there are two hypothesis, H_0 the area is not on fire and H_1 the area is on fire. Generally, the MAP detector has following condition to determine H_0:

$$p(H_0 \mid x) > p(H_1 \mid x) \tag{1}$$

Thus, suppose the determined results of the area state is according to the probability p that the area is not on fire, then there is able to solve the following optimal problem to determine the state of each node:

$$\begin{aligned} &\min -p * \frac{1}{\sqrt{2\pi\sigma^2} e^{\frac{(x(1)-20)_2}{2\sigma^2}}} - (1-p) * \frac{1}{\sqrt{2\pi\sigma^2} e^{\frac{(x(1)-80)_2}{2\sigma^2}}} \\ &\text{subject to } 0 \leq p \leq 1 \end{aligned} \tag{2}$$

Based on the p obtained, the state of sensor is able tot be determined.

In addition, the influence from neighbors need to be considered. In the report, it is assumed that if the sensor has one neighbor on fire (for sensors having three neighbors, the number should is two), the sensor area can be on fire of probability $\frac{1}{2}$. Then the fire region is determined due to both resulted Pr of equation 2 and influence from neighbors.

IV. Results and Analysis

In this section, the results generated by Matlab code are presented. The convergence speed and impact of transmission failures and sensor lost in both gossip algorithm are discussed. What's more, the influence of variance of temperature is also provided in fire detection.

i. Averaging Results

i.1 Performance of two gossip algorithm

For simplicity's sake, the variance of temperature is set as 1. The error is $\|x(k) - x_{ave}\|^2$ in dB. Then the plots showing the convergences of randomized gossip and greedy gossip for different iteration times are provided in Figure 2.

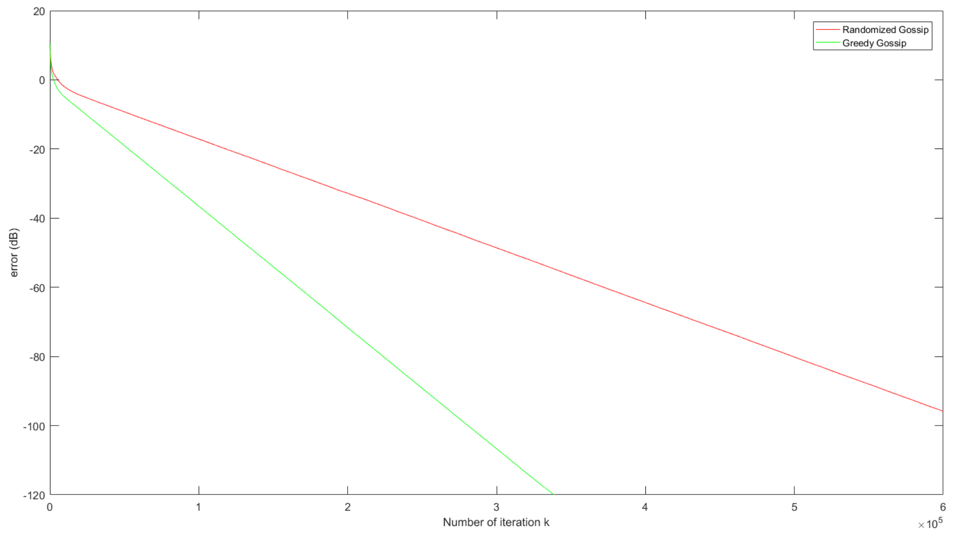


Figure 2: *Error in different number of iterations of two algorithms.*

It is clear in the plot that for larger iteration number, the error of both algorithms decreases and the value of each sensor in network is more close to the average value. Also, the convergence speed of greedy gossip with eavesdropping is faster than the randomized gossip at certain number of iteration.

i.2 Failure Cases

Once there are failures happening in the network, the performance of averaging algorithm will be affected. And the different failure cases is analyzed in this section.

(1) Transmission failure

If the transmission between a sensor and its neighbor failed, the node to be averaged is not able to get its neighbor's value and the averaging will not be carried out. But this condition does not have any influences on the transmission in the next time slot. In the next time slot, a new node will be chosen independently and make another transmission with one of its neighbor.

In the simulation, the probability of transmission failure is set as 0.1, 0.2 and 0.3 respectively. The results obtained by both algorithm is presented in Figure 3 and Figure 4.

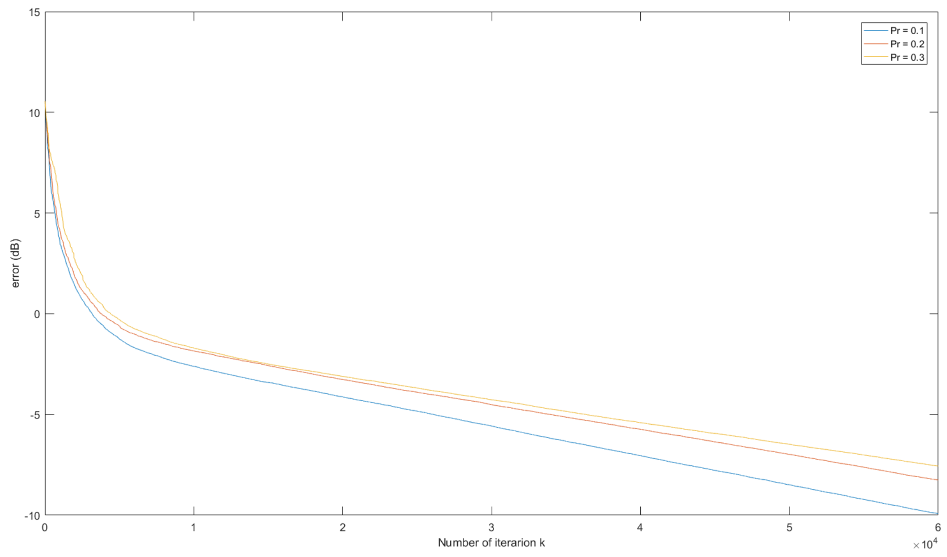


Figure 3: *Error in different number of iterations of different transmission failure probability by using randomized gossip algorithm.*

Based on the results, it can be seen that as the probability of transmission failure increases, the error in both algorithms increases slightly, which indicates that the transmission failure has impact on the averaging result but the influence is not so significant. Also, the performance of greedy gossip is better than the randomized gossip.

The results indicate that if the system has a huge number of iteration, it is necessary to maintain a lower Pr.

(2) Absence of sensors

If some sensors do not work anymore or are removed, the averaging resulted is changed definitely. Set the number of absent sensors to different values, the error in different number of iterations is obtained by using randomized gossip and greedy gossip, and the plots are shown in Figure 5 and Figure 6 respectively.

From Figure 5(a) and Figure 6(a), it can be obtained that as the number of absent sensors increases, the error increases. But in Figure 5(b) and Figure 6(b), with larger number of absent sensors, the results are different. When only a few part of sensors still stay, the error decreases as the number of kept sensor declines. This phenomenon is more clearly in Figure 7. IT may be because that if the number of kept sensors is small, the transmission times needed to reach the average value is small. So the error decreases. But in this case, not every sensors have neighbors, which means that most of the kept sensors may be separated, thus the error is still very high.

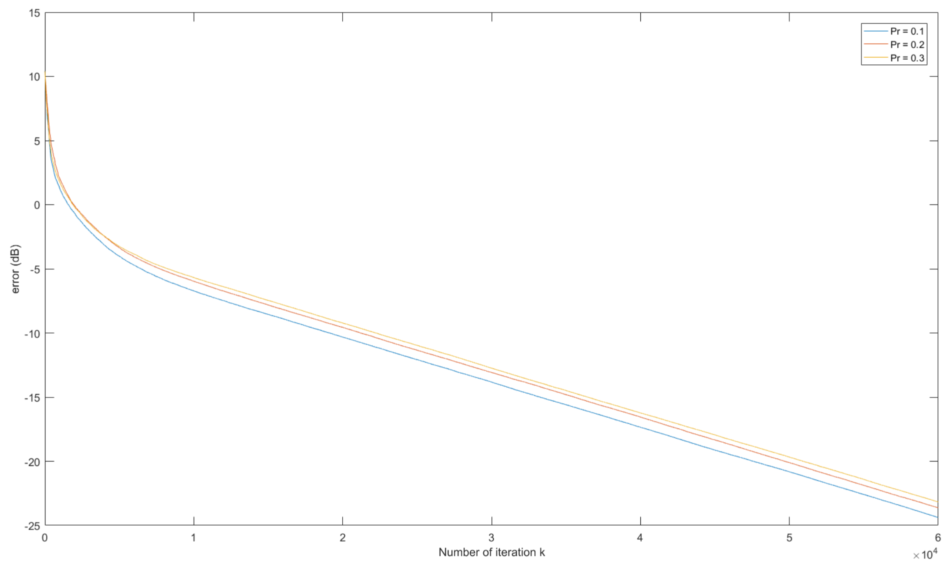


Figure 4: *Error in different number of iterations of different transmission failure probability by using greedy gossip algorithm.*

ii. Detection Results

In the fire detection experiment, the region on fire is assumed as sensor 26, 38, 39, 50, 51 and 62, which is shown in Figure 8.

Also, assume that if the probability p that the area is not on fire is smaller than 0.5, then the area is on fire and vice versa.

Thus there is following output Figure 9 that state the number of sensors whose area is on fire:

The detection result is the same with the assumed on number of sensor on fire. The reason is that the assumed area is in the center of the mat and on the edge of the plane. Thus the neighbors of the sensors in the fire region all have only on neighbor on fire. As assume in chapter 3, the sensor has only one neighbor on fire while it has three neighbors in total is determined as not on fire. Thus the results obtained is corresponding to the situation.

If the variances of the temperatures changes, result is provided in Figure 10.

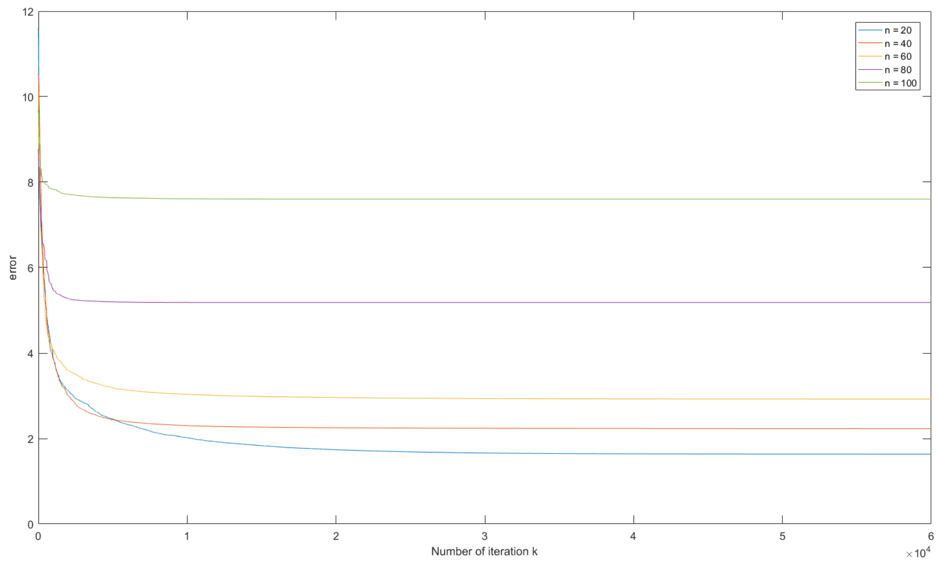
Thus it can be concluded from Figure 10 that as the variance increased, more sensors are determined as on fire area.

V. Conclusion

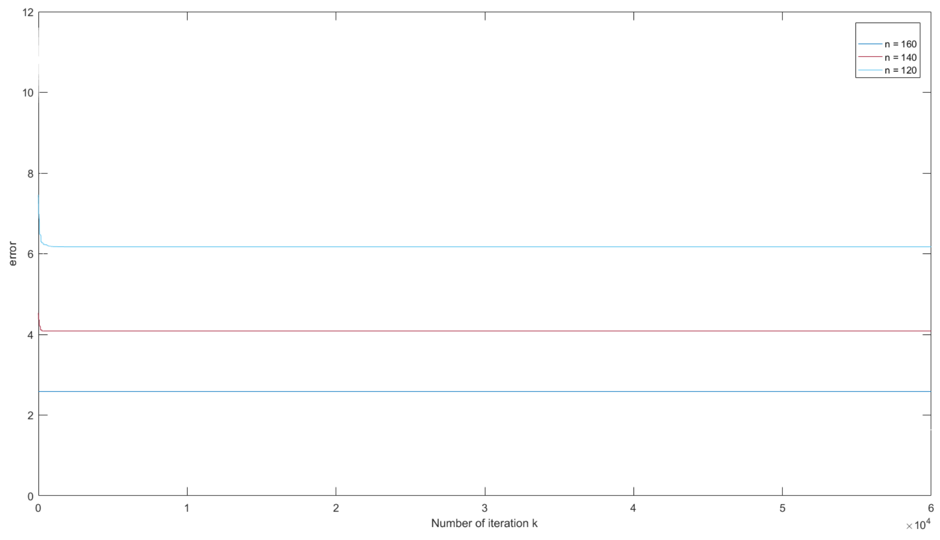
In the report, fire detection problem is solved in three parts when applying distributed signal processing technology.

Firstly, the sensor network topology is designed in hexagonal distribution as it has fewer sensors in the network and higher efficiency in convergence.

Then two gossip algorithms, randomized gossip and greedy gossip with eavesdropping are used to achieve averaging among the network. The results indicate that with larger number of iteration, the error decreases and greedy gossip has a higher convergence speed than randomized

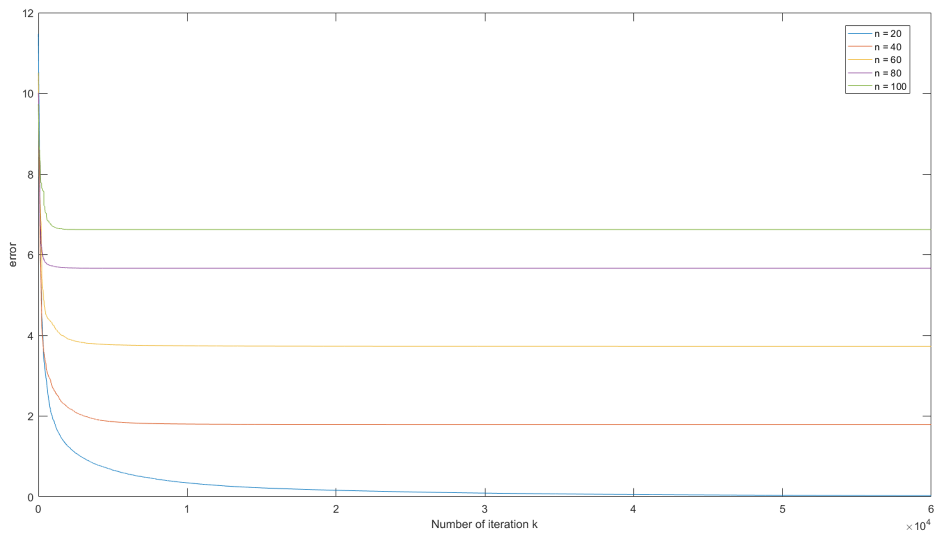


(a)

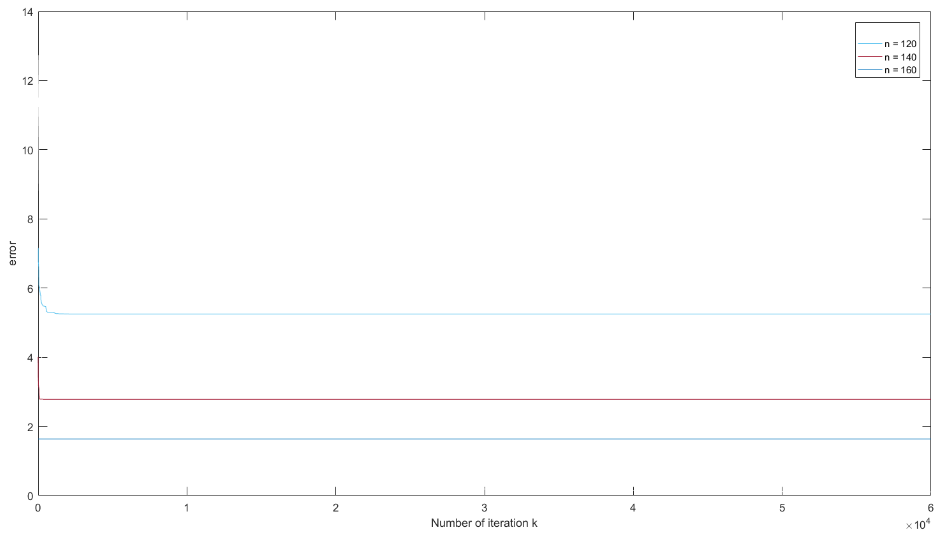


(b)

Figure 5: *Average error per transmission in different number of absent sensors by using randomized gossip algorithm. (a) number of absent sensors is 20, 40, 60, 80 and 100; (b) number of absent sensors is 120, 140 and 160.*



(a)



(b)

Figure 6: *Average error per transmission in different number of absent sensors by using greedy gossip algorithm. (a) number of absent sensors is 20, 40, 60, 80 and 100; (b) number of absent sensors is 120, 140 and 160.*

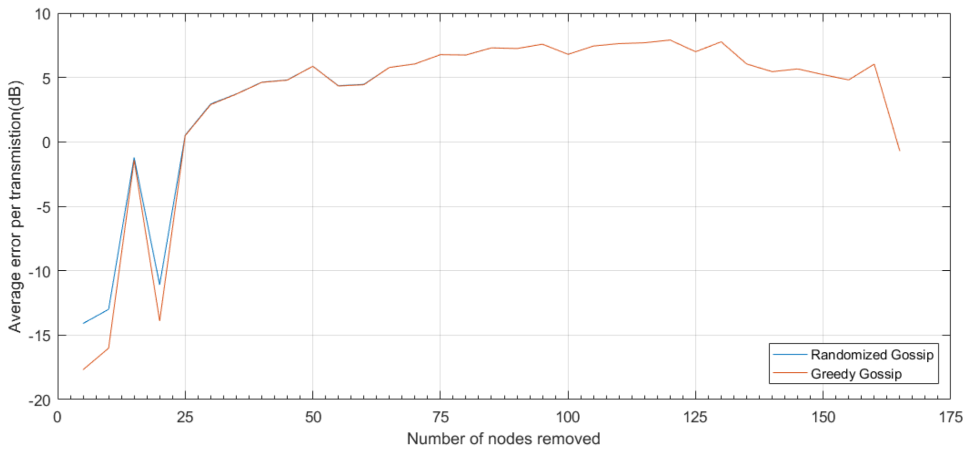


Figure 7: *Average error per transmission in different number of absent sensors by using both gossip algorithms.*

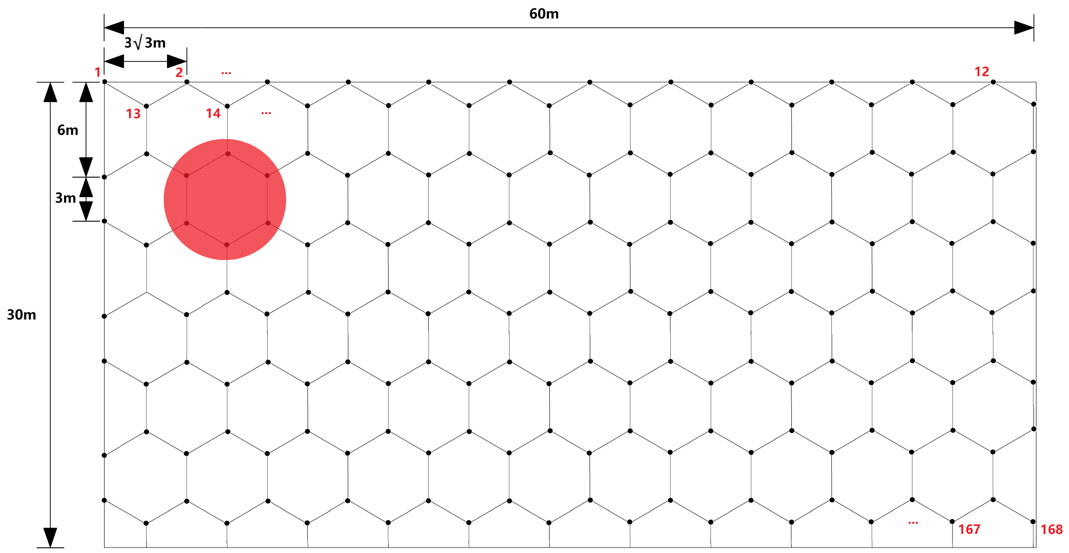
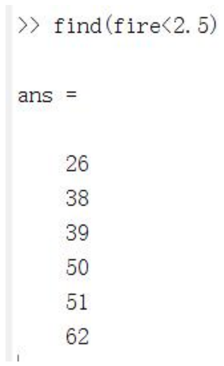


Figure 8: *Sensor area on fire.*



```
>> find(fire<2.5)

ans =

    26
    38
    39
    50
    51
    62
```

Figure 9: *Number of sensors whose area is on fire.*

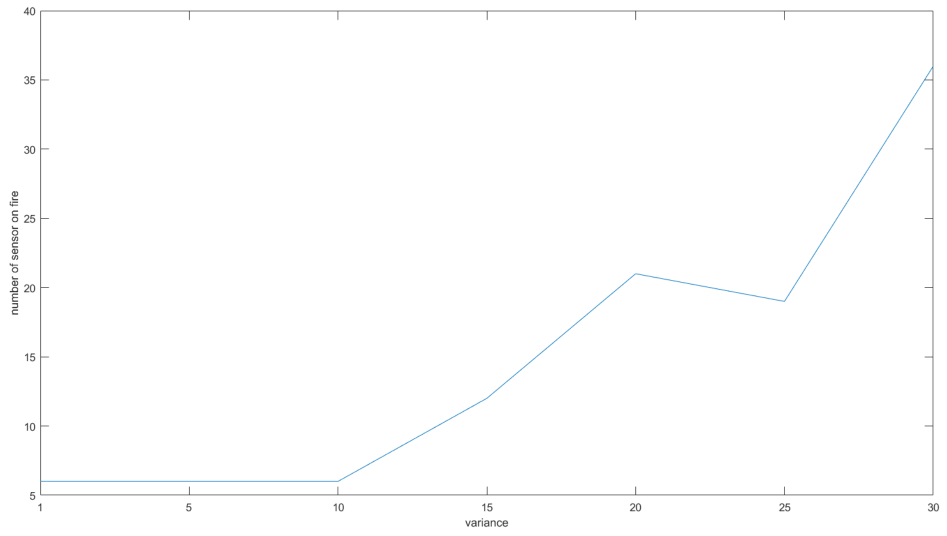


Figure 10: *Number of sensors determined as on fire in different variances.*

gossip under a certain number of iteration.

As for failure cases, if there are sensors absent, the tendency of error increases first and then declines after a roughly steady period. And the two algorithms have almost the same tendency. If the transmission failure exists, the probability of every transmission failure (Pr) influences the error obviously. As the Pr increases, the error goes up as well, and the differences of different Pr will increase at larger number of iteration.

The fire detection result in report is corresponding to the assumption which means that the method proposed in the report is effective. In addition, the variance influences the detection results a lot. If the variance of temperature increases, more sensors will be determined as on fire area.

A. Matlab code for performances of two gossip algorithms

```matlab
close all
clear all
clc

n = 168; %node number
d = zeros(n,1); %
err = inf;
err2 = inf;
k = 0;
kmax_n = 60000;
A = zeros(n,n);

%% node value initialization
a = 1:1:12; %x
b = 1:1:14; %y
x = zeros(n,1); %node value
for i = 1:1:n
    x(i) = normrnd(20,1);
end
x_ave = (sum(x)/n)*ones(n,1);

%% degree initialization
d = 3*ones(n,1);
bb = [1 14];
for i = 1:1:2
    for a = 1:1:12
        z = (bb(i)-1)*12+a;
        d(z) = 2;
    end
end
bb1 = [4 5 8 9 12 13];
bb2 = [2 3 6 7 10 11];
for i = 1:1:6
    a = 1;
    z = (bb1(i)-1)*12+a;
    d(z) = 2;
    a = 12;
    z = (bb2(i)-1)*12+a;
    d(z) = 2;
end
d(1)=1;
d(168)=1;

%% adjacency initialzation
% b=1, a=1
z_i = 1; z_o = 13;
A(z_i,z_o) = 1;
% b=1, a=2-12
b_i = 1;
for  a_i = 2:1:12;
```

```
    z_i = 12*(b_i-1)+a_i;
    z_o1 = 12*b_i+a_i-1;
    z_o2 = 12*b_i+a_i;
A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
end
% b=14, a=1-11
b_i = 14;
for  a_i = 1:1:11;
    z_i = 12*(b_i-1)+a_i;
    z_o1 = 12*b_i+a_i-24;
    z_o2 = 12*b_i+a_i-23;
A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
end
% b=14, a=12
z_i = 168; z_o = 156;
A(z_i,z_o) = 1;
%b=2, a=1-11
bbb = [2 6 10];
for i = 1:1:3
    for a_i = 1:1:11
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i-24;
    z_o2 = 12*bbb(i)+a_i-23;
    z_o3 = 12*bbb(i)+a_i;
    A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
A(z_i,z_o3) = 1;
    end
end
%b=2,a=12
a_i=12;
for i = 1:1:3
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i-24;
    z_o2 = 12*bbb(i)+a_i;
    A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
end
%b=3,a=1-11
bbb = [3 7 11];
for i = 1:1:3
    for a_i = 1:1:11
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i;
    z_o2 = 12*bbb(i)+a_i+1;
    z_o3 = 12*bbb(i)+a_i-24;
    A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
A(z_i,z_o3) = 1;
    end
```

```
end
%b=3,a=12
a_i=12;
for i = 1:1:3
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i-24;
    z_o2 = 12*bbb(i)+a_i;
    A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
end
%b=4,a=2-12
bbb = [4 8 12];
for i = 1:1:3
    for a_i = 2:1:12
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i-25;
    z_o2 = 12*bbb(i)+a_i-24;
    z_o3 = 12*bbb(i)+a_i;
    A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
A(z_i,z_o3) = 1;
    end
end
%b=4,a=1
a_i=1;
for i = 1:1:3
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i-24;
    z_o2 = 12*bbb(i)+a_i;
    A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
end
%b=5,a=2-12
bbb = [5 9 13];
for i = 1:1:3
    for a_i = 2:1:12
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i-24;
    z_o2 = 12*bbb(i)+a_i-1;
    z_o3 = 12*bbb(i)+a_i;
    A(z_i,z_o1) = 1;
A(z_i,z_o2) = 1;
A(z_i,z_o3) = 1;
    end
end
%b=5,a=1
a_i=1;
for i = 1:1:3
        z_i = (bbb(i)-1)*12+a_i;
        z_o1 = 12*bbb(i)+a_i-24;
    z_o2 = 12*bbb(i)+a_i;
    A(z_i,z_o1) = 1;
```

```
A(z_i,z_o2) = 1;
end

%% randomized gossip
xx = x;
errr = zeros(kmax_n,1);
kmax = kmax_n;
while (err(end) > 1e-12) && (k < kmax),
i = randi(n);
nnn = find(A(i,:)==1);
[V D] = size(nnn);
pos = unidrnd(D);
j = nnn(pos);
x([i j]) = .5*sum(x([i j]));
k = k+1;
err(k) = norm(x-x_ave);
end
plot(log10(err),'r')
hold on
xlabel('kmax');
ylabel('error (dB)');

%% greedy gossip with eavesdropping
x2 = xx;
k2 = 0;
while (err2(end) > 1e-12)&& (k2 < kmax),
i2 = randi(n);
nnn2 = find(A(i2,:)==1);
[V2 D2] = size(nnn2);
value = zeros(D2,1);
for m = 1:1:D2
    value(m) = abs(x2(i2)-x2(nnn2(m)));
end
pos2 = find(value == max(value));
[V3 D3] = size(pos2);
j2 = nnn2(pos2(1));
x2([i2 j2]) = .5*sum(x2([i2 j2]));
k2 = k2+1;
err2(k2) = norm(x2-x_ave);
end
plot(log10(err2),'g')
xlabel('kmax');
ylabel('error (dB)');
```

distributed_2_1.m

B. Matlab code for sensor absence case

```
close all
clear all
clc

n = 168; %node number
```

```
kmax = 60000;

%% node value initialization
a = 1:1:12;
b = 1:1:14;

%% degree initialization
load('d_value.mat');

%% a matrix new
load('Amatrix.mat');

sum_e1 = zeros(33,1);
sum_e2 = zeros(33,1);

for time_removenode = 1:1:33;
    number_removenode = time_removenode*5;
    p=randperm(n);
    A_new = A;
fff = ones(168,1);
x_new = zeros(168,1);
x_value = zeros(n-number_removenode,1); %node value
for i = 1:1:n-number_removenode
    x_value(i) = normrnd(20,1);
end

for w= 1:1:number_removenode
    A_new(p(w),:) = 0;
    A_new(:,p(w)) = 0;
    fff(p(w))=0;
end

x_ave = (sum(x_value)/(n-number_removenode))*fff;
x_new(fff~=0) = x_value;

%% randomized gossip
err1 = inf;
k1 = 0;
xx = x_new;
while (err1(end) > 1e-12) && (k1 < kmax),
i = randi(n); %chosen average target
n_set1 = find(A_new(i,:)==1);%neighbor set
n_number1 = length(n_set1);%neighbor number
if n_number1 == 0
else
n_pos1 = unidrnd(n_number1);%chosen neighbor position in set
j = n_set1(n_pos1);%chosen neighbor
x_new([i j]) = .5*sum(x_new([i j])); %averaging
end
k1 = k1+1;
err1(k1) = norm(x_new-x_ave);%k-th average error
end
```

```

sum_e1(time_removenode) = sum(err1);

%% greedy gossip with eavesdropping
x_new2 = xx;
err2 = inf;
k2 = 0;
while (err2(end) > 1e-12) && (k2 < kmax),
i2 = randi(n);
n_set2 = find(A_new(i2,:)==1);
n_number2 = length(n_set2);
if n_number2== 0
else
value = zeros(n_number2,1);
for m = 1:1:n_number2
    value(m) = abs(x_new2(i2)-x_new2(n_set2(m)));
end
n_pos2 = find(value == max(value));
j2 = n_set2(n_pos2(1));
x_new2([i2 j2]) = .5*sum(x_new2([i2 j2]));
end
k2 = k2+1;
err2(k2) = norm(x_new2-x_ave);

end

sum_e2(time_removenode) = sum(err2);
end
plot(10*log10(sum_e1/kmax));
hold on
plot(10*log10(sum_e2/kmax));
xlabel('Number of nodes removed')
ylabel('Average error per transmistion(dB)')
```

distributed_2_22.m

C. Matlab code for transmission failure case

```
close all
clear all
clc

n = 168; %node number
d = zeros(n,1); %
err = inf;
err2 = inf;
k = 0;
kmax_n = 60000;
A = zeros(n,n);

%% node value initialization
a = 1:1:12; %x
b = 1:1:14; %y
```

```
x = zeros(n,1); %node value
for i = 1:1:n
    x(i) = normrnd(20,1);
end
x_ave = (sum(x)/n)*ones(n,1);

%% degree initialization
load('d_value.mat');

%% adjacency initialzation
load('Amatrix.mat');

%% randomized gossip
errr = zeros(kmax_n,1);
cc = [1 2 3];
for cccc = 1:1:3
%      xx = x;
%      err = inf;
    kmax = kmax_n;
%      k1 = 0;
% while (err(end) > 1e-12) && (k < kmax),
% i = randi(n);
% nnn = find(A(i,:)==1);
% [V D] = size(nnn);
% pos = unidrnd(D);
% j = nnn(pos);
% % j = nei(randi(d(i)));
% ccc = randperm(10);
% if ccc(1)>cc(cccc)
%      xx([i j]) = .5*sum(xx([i j]));
% else
% end
% k = k+1;
% err(k) = norm(xx-x_ave);
% end
% % errr(kmax) = norm(x-x_ave);
% % end
%
% plot(10*log10(err))
% hold on
% xlabel('kmax');
% ylabel('error (dB)');

%% greedy gossip with eavesdropping
x2 = x;
err2 = inf;
k2 = 0;

while (err2(end) > 1e-12)&& (k2 < kmax),
i2 = randi(n);
nnn2 = find(A(i2,:)==1);
[V2 D2] = size(nnn2);
```

```
value = zeros(D2,1);
ccc = randperm(10);
if ccc(1)>cc(cccc)
    for m = 1:1:D2
    value(m) = abs(x2(i2)-x2(nnn2(m)));
end
pos2 = find(value == max(value));
[V3 D3] = size(pos2);
if D3 == 1
    j2 = nnn2(pos2);
else
    j2 = nnn2(pos2(unidrnd(D3)));
end
j2 = nnn2(pos2(1));
x2([i2 j2]) = .5*sum(x2([i2 j2]));
k2 = k2+1;
err2(k2) = norm(x2-x_ave);
else
end
end
plot(10*log10(err2))
hold on
xlabel('Number of iteration k');
ylabel('error (dB)');
end
```

distributed_2_3.m

D. Matlab code for obtaining probability of sensor not on fire

```
close all
clear all
clc

n = 168; %node number
x = zeros(n,1); %node value
for i = 1:1:n
    x(i) = normrnd(20,1);
end

cvx_begin

variable p;
minimize( -(p/sqrt(2*pi))*exp(-(x(1)-20)^2/2)-((1-p)/sqrt(2*pi))*exp(-(x(1)
    -80)^2/2) );
subject to
0 <= p;
p <= 1;

cvx_end
```

convex.m

E. Matlab code for fire region detection

```
close all
clear all
clc

n = 168; %node number
d = zeros(n,1); %
err = inf;
err2 = inf;
k = 0;
kmax_n = 600000;
A = zeros(n,n);

%% node value initialization
a = 1:1:12; %x
b = 1:1:14; %y
x = zeros(n,1); %node value
for i = 1:1:n
    x(i) = normrnd(20,1);
end

%% choose node in fire area
n_chose = [26,38,39,50,51,62];
for ss = 1:1:6
    x(n_chose(ss)) = normrnd(80,1);
end

%% degree initialization
load('d_value.mat');

%% a matrix
load('Amatrix.mat');

%% detection

%detect original fire node
p_x = zeros(168,1);

for sss = 1:1:168

    cvx_begin
variable p;
minimize( -(p/sqrt(2*pi))*exp(-(x(sss)-20)^2/2)-((1-p)/sqrt(2*pi))*exp(-(x(
    sss)-80)^2/2) );
subject to
0 <= p;
p <= 1;
cvx_end

p_x(sss) = p;
end
```

```

%detect fire node due to neighbor
fire_n = ones(168,1);
for ssss = 1:1:168
  nnn = find(A(ssss,:)==1);
  length_n = length(nnn);
  if length_n == 1
      if p_x(nnn)>=0.5
      else
          fire_n(ssss) = unidrnd(2)-1;
      end
  elseif length_n == 2
      if p_x(nnn(1))>=0.5 && p_x(nnn(2))>=0.5
      elseif p_x(nnn(1))>=0.5 || p_x(nnn(2))>=0.5
          fire_n(ssss) = unidrnd(2)-1;
      else
          fire_n(ssss) = 0;
      end
  else
      sum_p = p_x(nnn(1))+p_x(nnn(2))+p_x(nnn(3));
      if sum_p>1.5
      elseif sum_p<=0.5
          fire_n(ssss) = 0;
      else 0.5<sum_p<=1.5
          fire_n(ssss) = unidrnd(2)-1;
      end
  end
end

  fire = 2*fire_n+p_x;
  find(fire<2.5)%output fire node number
```

distributed_3.m

F. Matlab code for impact of variance in fire detection

```
close all
clear all
clc

n = 168; %node number
d = zeros(n,1); %
err = inf;
err2 = inf;
k = 0;
kmax_n = 600000;
A = zeros(n,n);
firenumber = zeros(7,1);
v = [1 5 10 15 20 25 30];
for vv = 1:1:7
%% node value initialization
a = 1:1:12; %x
b = 1:1:14; %y
```

```matlab
x = zeros(n,1); %node value
for i = 1:1:n
    x(i) = normrnd(20,v(vv));
end

%% choose node in fire area
n_chose = [26,38,39,50,51,62];
for ss = 1:1:6
    x(n_chose(ss)) = normrnd(80,v(vv));
end

%% degree initialization
load('d_value.mat');

%% a matrix
load('Amatrix.mat');

%% detection

%detect original fire node
p_x = zeros(168,1);

for sss = 1:1:168

    cvx_begin
variable p;
minimize( -(p/sqrt(2*pi*v(vv)^2))*exp(-(x(sss)-20)^2/(2*v(vv)^2))-((1-p)/
    sqrt(2*pi*v(vv)^2))*exp(-(x(sss)-80)^2/(2*v(vv)^2)) );
subject to
0 <= p;
p <= 1;
cvx_end

p_x(sss) = p;
end

%detect fire node due to neighbor
fire_n = ones(168,1);
for ssss = 1:1:168
  nnn = find(A(ssss,:)==1);
  length_n = length(nnn);
  if length_n == 1
      if p_x(nnn)>=0.5
      else
          fire_n(ssss) = unidrnd(2)-1;
      end
  elseif length_n == 2
      if p_x(nnn(1))>=0.5 && p_x(nnn(2))>=0.5
      elseif p_x(nnn(1))>=0.5 || p_x(nnn(2))>=0.5
          fire_n(ssss) = unidrnd(2)-1;
      else
          fire_n(ssss) = 0;
```

```
        end
    else
        sum_p = p_x(nnn(1))+p_x(nnn(2))+p_x(nnn(3));
        if sum_p>1.5
        elseif sum_p<=0.5
            fire_n(ssss) = 0;
        else 0.5<sum_p<=1.5
            fire_n(ssss) = unidrnd(2)-1;
        end
    end
end

  fire = 2*fire_n+p_x;
 vvv =  find(fire<2.5);%output fire node number
 firenumber(vv) = length(vvv);
end
   plot(firenumber);
   xlabel('variance');
   ylabel('number of sensor on fire')
```

distributed_3_1.m

References

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Method of Multipliers, Foundations and Trends in Machine Learning, vol. 3, no. 1, pp.1-122, 2011.