

# **LAPORAN FINAL PROJECT**

## **TOPIK DALAM DATA MINING A**



Disusun oleh :

Reza Presetya Prayogo    5116201009

Ari Effendi                      5116201016

Addien Haniefardy        5116201049

**Jurusan Teknik Informatika**  
**Fakultas Teknologi Informasi**  
**Institut Teknologi Sepuluh Nopember Surabaya**

**2017**

## 1. Paper Acuan

**“Using the Stability of Objects to Determine the Number of Clusters in Datasets”**

## 2. Latar Belakang

Clustering merupakan suatu cara yang digunakan untuk membagi sejumlah data menjadi kelompok-kelompok data atau mengelompokkan suatu objek dengan objek yang lain berdasarkan similaritasnya. Semakin similar suatu objek/data dengan objek/data yang lain, semakin besar pula kemungkinan kedua objek tersebut tergabung dalam satu cluster (kelompok). Nilai similaritas dihitung menggunakan fitur-fitur/atribut-atribut yang terdapat pada setiap objek/data.

Dalam melakukan clustering, terdapat sejumlah metode yang bisa digunakan, diantaranya menggunakan metode K-Means. K-Means merupakan salah satu algoritma yang sering digunakan dalam klusterisasi karena mudah diimplementasikan dan relatif cepat. Huruf K pada K-Means menunjukkan jumlah kelompok yang ingin dibuat. Pada perkembangannya, nilai K menjadi nilai yang sangat penting untuk ditentukan dalam menentukan cluster yang paling optimal.

Untuk menentukan nilai K, banyak sekali pendekatan yang bisa dilakukan, diantaranya menggunakan algoritma Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), dan lain sebagainya. Pada paper ini, akan dikembangkan algoritma penentuan nilai K pada K-Means menggunakan nilai stabilitas objek. Nilai stabilitas objek sendiri dihitung menggunakan metode Silhouette.

## 3. Tujuan

Implementasi paper dalam melakukan optimasi nilai K pada K-Means menggunakan nilai stabilitas objek dengan metode Silhouette

## 4. Dataset

Pada final project kali ini, kami menggunakan dataset Zoo dengan 101 objek, 17 fitur, dan nilai K sama dengan 7 sebagai bentuk klaster optimal. Pada gambar pertama, bisa dilihat

```
1 (41) aardvark, antelope, bear, boar, buffalo, calf,
      cavy, cheetah, deer, dolphin, elephant,
      fruitbat, giraffe, girl, goat, gorilla, hamster,
      hare, leopard, lion, lynx, mink, mole, mongoose,
      opossum, oryx, platypus, polecat, pony,
      porpoise, puma, pussycat, raccoon, reindeer,
      seal, sealion, squirrel, vampire, vole, wallaby, wolf
2 (20) chicken, crow, dove, duck, flamingo, gull, hawk,
      kiwi, lark, ostrich, parakeet, penguin, pheasant,
      rhea, skimmer, skua, sparrow, swan, vulture, wren
3 (5)  pitviper, seasnake, slowworm, tortoise, tuatara
4 (13) bass, carp, catfish, chub, dogfish, haddock,
      herring, pike, piranha, seahorse, sole, stingray, tuna
5 (4)  frog, frog, newt, toad
6 (8)  flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp
7 (10) clam, crab, crayfish, lobster, octopus,
      scorpion, seawasp, slug, starfish, worm
```

*Gambar 1 Dataset Zoo - 101 Objek yang terbagi ke dalam 7 klaster*

101 objek yang tersebar ke dalam 7 kluster. Pada gambar kedua, bisa dilihat fitur (2-18) yang dimiliki oleh setiap objek beserta tipe nilainya.

Attribute Information: (name of attribute and type of value domain)	
1. animal name:	Unique for each instance
2. hair	Boolean
3. feathers	Boolean
4. eggs	Boolean
5. milk	Boolean
6. airborne	Boolean
7. aquatic	Boolean
8. predator	Boolean
9. toothed	Boolean
10. backbone	Boolean
11. breathes	Boolean
12. venomous	Boolean
13. fins	Boolean
14. legs	Numeric (set of values: {0,2,4,5,6,8})
15. tail	Boolean
16. domestic	Boolean
17. catsize	Boolean
18. type	Numeric (integer values in range [1,7])

Gambar 2 Fitur-fitur pada dataset Zoo beserta tipe nilainya

## 5. Mekanisme Algoritma

Sebelum masuk pada algoritma, terlebih dahulu kita definisikan nilai  $R$ ,  $K_{\min}$  dan  $K_{\max}$ .  $R$  merupakan jumlah partisi yang ingin dibuat. Jumlah partisi ini akan menentukan nilai stabilitas dari suatu objek.  $K_{\min}$  merupakan nilai  $K$  minimal dan  $K_{\max}$  merupakan nilai  $K$  maksimal yang diinginkan. Solusi nilai  $K$  yang didapatkan akan berada pada rentan  $K_{\min} \leq K \leq K_{\max}$ . Pseudocode algoritma ditunjukkan pada gambar xx.

```

For  $K = K_{\min}$  to  $K_{\max}$  do
  For  $r = 1$  to  $R$  do
    Generate a random starting partition ( $RP_r$ ) of objects of  $\Omega$  into  $K$  non-
    empty classes
    Execute K-Means using  $RP_r$  as input to get partition  $Pr$ 
  End
  For  $i = 1$  to  $N$  do
    Compute the singleton support  $PS_i$ 
    For  $j = i + 1$  to  $N$  do
      Compute the pairwise support  $PS_{ij}$ 
    End
  End
  For  $i = 1$  to  $N$  do
    Compute the individual stability index  $ST_{\text{approx}}(i, K)$ 
  End
  Compute the global stability index,  $ST_{\text{global}}(K)$  for the case of  $K$  classes
End
Find the maximum value of  $ST_{\text{global}}(K)$  over all tested of  $K$ 
The optimal number of classes,  $K_{\text{opt}}$ , will correspond to the maximum of  $ST_{\text{global}}(K)$ 

```

Untuk penjelasan lebih detail, kami berikan contoh dataset Zoo ( $N = 101$ ) dengan nilai  $R$  sama dengan 1000,  $K_{\min}$  sama dengan 2, dan  $K_{\max}$  sama dengan 10. Tujuan dari algoritma

ini adalah untuk mendapatkan nilai K yang paling optimal ( $K_{opt}$ ). Pada perulangan pertama ( $K = 2$ ), dilakukan sub perulangan pertama untuk membuat R partisi. Jadi, dihasilkan 1000 partisi dengan setiap partisi terdiri dari 101 objek. Setelah itu, setiap partisi kemudian dilakukan pengklasteran menggunakan K-Means dengan nilai K sama dengan 2. Kemudian pada sub perulangan kedua, dilakukan 3 penghitungan. Pertama, penghitungan nilai Silhouette setiap objek pada semua partisi. Kedua, penghitungan nilai  $PS_i$  menggunakan persamaan 1. Ketiga, penghitungan nilai  $PS_{ij}$  menggunakan persamaan 2.

$$PS_i = \frac{\sum_{r=1}^R S_{r,i}}{\sum_{r=1}^R S_r} \quad (1)$$

$$PS_{ij} = \frac{\sum_{r=1}^R S_{r,ij}}{\sum_{r=1}^R S_r} \quad (2)$$

Dengan  $PS_i$  merupakan nilai stabilitas objek ke-i pada semua partisi. Dan  $PS_{ij}$  merupakan nilai stabilitas objek ke-i terhadap objek ke-j pada semua partisi. Lalu  $S_r$  merupakan nilai rata-rata dari penjumlahan nilai Silhouette semua objek pada satu partisi. Nilai  $PS_i$  dan  $PS_{ij}$  ini kemudian akan digunakan pada sub perulangan ketiga untuk menghitung nilai  $ST_{approx}(i,K)$  menggunakan persamaan 3.

$$\begin{aligned} ST_{approx}(i) = & \frac{1}{N} \sum_{j=1(j \neq i)}^N \max \left( \frac{K}{K-1} * \left( PS_{ij} - \frac{1}{K} \right); K * \left( \frac{1}{K} - PS_{ij} \right) \right) \\ & + \frac{1}{N} \max \left( \frac{K^{N-1}}{K^{N-1} - (K-1)^{N-1}} * \left( PS_i - \frac{(K-1)^{N-1}}{K^{N-1}} \right); \frac{K^{N-1}}{(K-1)^{N-1}} \right. \\ & \left. * \left( \frac{(K-1)^{N-1}}{K^{N-1}} - PS_i \right) \right) \end{aligned} \quad (3)$$

Setelah nilai  $ST_{approx}(i,K)$  didapatkan, dilakukan penghitungan  $ST_{global}(K)$  menggunakan persamaan 4.

$$ST_{global} = \frac{1}{N} \sum_{i=1}^N ST(i) \quad (4)$$

Untuk perulangan kedua ( $K = 3$ ), perulangan ketiga ( $K = 4$ ), dan seterusnya lakukan hal yang sama sampai mendapatkan nilai  $ST_{global}(K)$ . Setelah kita memperoleh nilai  $ST_{global}$  untuk setiap K, langkah selanjutnya adalah mendapatkan nilai  $ST_{global}$  tertinggi dari nilai-nilai  $ST_{global}$  tersebut. K dengan nilai  $ST_{global}$  tertinggi merupakan  $K_{opt}$ .

## 6. Uji Coba

### 1. Mendapatkan partisi $P_r$ menggunakan K-means

```

load('fdata.mat');
kmin=2; %Jumlah K minimal
kmax=4; %Jumlah K maksimal
[N,Nfitur] = size(fdata);
N=length(fdata); %Jumlah objek
R=3; %Random start K-means
for k=kmin:kmax
    for r=1:R
        temp_kmeans = kmeans(fdata,k);
        for RPr=1:N
            for SRcount=1:N
                for NfiturCount=1:Nfitur
                    Pr{1,k}{RPr,r}(SRcount,NfiturCount) = fdata(SRcount,NfiturCount);
                end
            end
            for kmeanscount=1:N
                Pr{1,k}{RPr,r}(kmeanscount,Nfitur+1) = temp_kmeans(kmeanscount,1);
            end
            for silhouetteCount=1:N
                temp_silhouette = silhouette(Pr{1,k}{1,r}(:,1:Nfitur), Pr{1,k}{1,r}(:,Nfitur+1),'SqEuclidean');
            end
            for count_tmp_silh=1:N
                Pr{1,k}{RPr,r}(count_tmp_silh,Nfitur+2) = temp_silhouette(count_tmp_silh,1);
            end
        end
    end
end

```

Activate Windows  
Go to Settings to activate Windows.

**Gambar 6.1 Partisi Pr**

2. a. Mendapatkan singleton support PS<sub>i</sub> menggunakan Eq. (2)

```

kmin=2;
kmax=4;
R=3;
[N,Nfitur] = size(fdata);
N=length(fdata); %Jumlah objek
for k=kmin:kmax
    for count_tmp_SRi=1:N
        SRi_tmp = 0;
        SRi{count_tmp_SRi,1} = 0;
        for r=1:R
            SRi_tmp = SRi_tmp+Pr{1,k}{1,r}(count_tmp_SRi,Nfitur+2);
        end
        SRi{count_tmp_SRi,k} = SRi_tmp;
    end
end

```

**Gambar 6.2.a Partition Score SRI**

```

kmin=2;
kmax=4;
R=3;
[N,Nfitur] = size(fdata);
N=length(fdata); %Jumlah objek
for k=kmin:kmax
    SR_total = 0;
    for r=1:R
        SR_tmp = 0;
        for count_tmp_SR=1:N
            SR_tmp = SR_tmp+Pr{1,k}{1,r}(count_tmp_SR,Nfitur+2);
        end
        SR(r,k) = SR_tmp/N;
        SR_total = SR_total+SR_tmp/N;
    end
    SR_sum(1,k) = SR_total;
end

```

**Gambar 6.2.a Partition Score Sr**

```

kmin=2;
kmax=4;
R=3;
[N,Nfitur] = size(fdata);
N=length(fdata);
for k=kmin:kmax
    for count_PSi=1:N
        PSi(count_PSi,k) = SRi(count_PSi,k)/SR(1,k);
    end
end

```

**Gambar 6.2.a singleton support PSi**

b. Mendapatkan pairwise support  $PS_{ij}$  menggunakan Eq. (1)

```

kmin=2;
kmax=4;
R=3;
[N,Nfitur] = size(fdata);
N=length(fdata);
for k=kmin:kmax
    for r=1:R
        for i=1:N+1
            for j=i+1:N
                if(Pr{1,k}{1,r}(i,Nfitur+1)==Pr{1,k}{1,r}(j,Nfitur+1))
                    SRij{r,k}(i,j) = SR(r,k);
                else
                    SRij{r,k}(i,j) = 0;
                end
            end
        end
    end
end

```

**Gambar 6.2.b partition score SRij**

```

kmin=2;
kmax=4;
R=3;
[N,Nfitur] = size(fdata);
N=length(fdata);
for k=kmin:kmax
    for i=1:N+1
        for j=i+1:N
            tmp_PSi_j = 0;
            for r=1:R
                tmp_PSi_j = tmp_PSi_j + SRij{r,k}(i,j);
            end
            PSij{l,k}(i,j) = tmp_PSi_j/SR_sum(l,k);
        end
    end
end
end

```

**Gambar 6.2.b pairwise support PSij**

3. Mendapatkan individual stability index  $ST(i,K)$  atau approximate variant  $ST_{approx}(i,K)$  menggunakan Eq. (9) dan (10).

```

kmin=2;
kmax=4;
R=3;
[N,Nfitur] = size(fdata);
N=length(fdata);
for k=kmin:kmax
    for i=1:N
        hasil1 = 0;
        hasil2 = 0;
        for j=i+1:N
            j1 = (k/(k-1))*(PSij{l,k}(i,j)-(1/k));
            j2 = k*((1/k)-PSij{l,k}(i,j));
            if (j1 > j2)
                hasil1 = hasil1 + j1;
            else
                hasil1 = hasil1 + j2;
            end
        end
        hasil1 = hasil1;
        i1 = (k.^(N-1)/(k.^(N-1)-(k-1).^(N-1)))*(PSi(i,k)-((k-1).^(N-1)/k.^(N-1)));
        i2 = (k.^(N-1)/(k-1).^(N-1))*(((k-1).^(N-1)/k.^(N-1))-PSi(i,k));
        if (i1 > i2)
            hasil2 = i1;
        else
            hasil2 = i2;
        end
        STapprox(i,k) = (hasil1 + hasil2)/N;
    end
end

```

**Gambar 6.3 approximate variant  $ST_{approx}$**

4. Mendapatkan global stability index menggunakan Eq. (12).

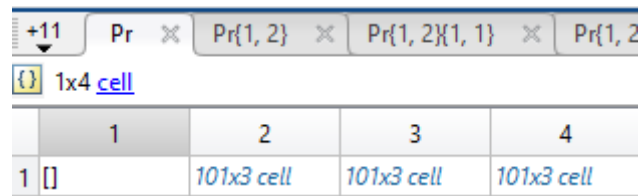
```

kmin=2;
kmax=4;
R=3;
[N,Nfitur] = size(fdata);
N=length(fdata);
for k=kmin:kmax
    tmp_STglobal = 0;
    for i=1:N
        tmp_STglobal = tmp_STglobal + STapprox(i,k);
    end
    STglobal{l,k} = tmp_STglobal / N;
end
end

```

**Gambar 6.4 global stability index  $ST_{global}$**

## 7. Hasil



	1	2	3	4
1		101x3 cell	101x3 cell	101x3 cell

**Gambar 7.1 partisi kmin=2, kmax=4**



	1	2	3
10	101x19 dou...	101x19 dou...	101x19 dou...
11	101x19 dou...	101x19 dou...	101x19 dou...
12	101x19 dou...	101x19 dou...	101x19 dou...
13	101x19 dou...	101x19 dou...	101x19 dou...
14	101x19 dou...	101x19 dou...	101x19 dou...
15	101x19 dou...	101x19 dou...	101x19 dou...
16	101x19 dou...	101x19 dou...	101x19 dou...
17	101x19 dou...	101x19 dou...	101x19 dou...
18	101x19 dou...	101x19 dou...	101x19 dou...
19	101x19 dou...	101x19 dou...	101x19 dou...
20	101x19 dou...	101x19 dou...	101x19 dou...
21	101x19 dou...	101x19 dou...	101x19 dou...
22	101x19 dou...	101x19 dou...	101x19 dou...
23	101x19 dou...	101x19 dou...	101x19 dou...
24	101x19 dou...	101x19 dou...	101x19 dou...
25	101x19 dou...	101x19 dou...	101x19 dou...
26	101x19 dou...	101x19 dou...	101x19 dou...
27	101x19 dou...	101x19 dou...	101x19 dou...
28	101x19 dou...	101x19 dou...	101x19 dou...
29	101x19 dou...	101x19 dou...	101x19 dou...
30	101x19 dou...	101x19 dou...	101x19 dou...

**Gambar 7.2 random start R=3**



+10	Pr(1, 2)	Pr(1, 2)(1, 1)	Pr(1, 2)(1, 2)	SR	SR_sum	SRi	SRij	SRij(1, 2)	Pr(1, 2)(18, 1)	Pr(1, 2)(20, 1)	SRij(2, 2)	
Pr(1, 2)(1, 2)												
6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	1	1	1	1	0	0	4	0	0	1	1
2	0	0	1	1	1	0	0	4	1	0	1	1
3	1	1	1	1	0	0	1	0	1	0	0	4
4	0	1	1	1	1	0	0	4	0	0	1	1
5	0	1	1	1	1	0	0	4	1	0	1	1
6	0	0	1	1	1	0	0	4	1	0	1	1
7	0	0	1	1	1	0	0	4	1	1	1	1
8	1	0	1	1	0	0	1	0	1	1	0	4
9	1	1	1	1	0	0	1	0	1	0	0	4
10	0	0	1	1	1	0	0	4	0	1	0	1
11	0	1	1	1	1	0	0	4	1	0	1	1
12	0	0	0	1	1	0	0	2	1	1	0	2
13	1	1	1	1	0	0	1	0	1	0	0	4
14	0	1	0	0	0	0	0	0	0	0	0	7
15	1	1	0	0	0	0	0	4	0	0	0	7
16	1	1	0	0	0	0	0	6	0	0	0	7
17	0	1	0	1	1	0	0	2	1	0	0	2
18	0	0	1	1	1	0	0	4	1	0	1	1
19	1	1	1	1	0	0	1	0	1	0	1	4
20	1	1	1	1	1	0	1	0	1	0	1	1
21	0	0	0	1	1	0	0	2	1	0	0	2

**Gambar 7.3 partisi Pr menggunakan Kmeans**

Kolom ke-1 sampai 17 merupakan fitur dari Dataset, sedangkan kolom ke-18 merupakan index cluster yang terbentuk menggunakan metode kmeans

+10	Pr{1, 2}	Pr{1, 2}{1, 1}	Pr{1, 2}{1, 2}	SR	SR_sum
	3x4 double				
	1	2	3	4	5
1	0		0.6956	0.6403	
2	0	0.6589	0.6784	0.5206	
3	0	0.6589	0.3112	0.6615	

**Gambar 7.4 partition score Sr**

$\Pr\{1, 2\} \{1, 1\}$	$\Pr\{1, 2\} \{1, 2\}$	SR	SR_sum	
2	3	4	5	6
1.8687	1.6852	1.8225		

**Gambar 7.5 sum partition score Sr**

+10	SR	SR_sum	SRi	SRij	SRij
101x4 cell					
	1	2	3	4	
1	0	2.1995	1.4342	2.3921	
2	0	2.2685	1.4307	2.5379	
3	0	0.8941	2.3169	1.7727	
4	0	2.1995	1.4342	2.3921	
5	0	2.2733	1.4356	2.5358	
6	0	2.2685	1.4307	2.5379	
7	0	2.2347	1.4160	2.4378	
8	0	0.8672	2.1431	1.6458	
9	0	0.8941	2.3169	1.7727	
10	0	2.1357	1.3853	2.2074	
11	0	2.2733	1.4356	2.5358	
12	0	2.1224	1.7590	1.3783	
13	0	0.8941	2.3169	1.7727	
14	0	0.9541	1.8026	1.5962	
15	0	2.0396	1.2975	2.0825	
16	0	1.9857	2.0401	2.5967	
17	0	2.1655	1.7768	1.5058	
18	0	2.2685	1.4307	2.5379	
19	0	0.8893	2.2331	1.6124	
20	0	2.1742	0.5259	1.6236	
21	0	2.1224	1.7590	1.3783	

**Gambar 7.6 partition score  $Sr_i$**

	1	2	3	4	
1	0	3.9928	2.0619	3.7359	
2	0	4.1179	2.0569	3.9636	
3	0	1.6230	3.3310	2.7686	
4	0	3.9928	2.0619	3.7359	
5	0	4.1268	2.0639	3.9604	
6	0	4.1179	2.0569	3.9636	
7	0	4.0566	2.0358	3.8074	
8	0	1.5743	3.0812	2.5704	
9	0	1.6230	3.3310	2.7686	
10	0	3.8769	1.9917	3.4475	
11	0	4.1268	2.0639	3.9604	
12	0	3.8528	2.5290	2.1527	
13	0	1.6230	3.3310	2.7686	
14	0	1.7321	2.5916	2.4930	
15	0	3.7024	1.8653	3.2525	
16	0	3.6046	2.9331	4.0555	
17	0	3.9310	2.5544	2.3517	
18	0	4.1179	2.0569	3.9636	
19	0	1.6143	3.2105	2.5182	
20	0	3.9469	0.7561	2.5358	
21	0	3.8528	2.5290	2.1527	

**Gambar 7.7 singleton support  $PS_i$**

+10

SR

SR\_sum

SRi

SRij

SRi

3x4 cell

	1	2	3	4
1		100x101 da...	100x101 da...	100x101 da...
2		100x101 da...	100x101 da...	100x101 da...
3		100x101 da...	100x101 da...	100x101 da...

+4

SR

SR\_sum

SRi

SRij

SRij(2, 2)

PSi

PSij

PSij(1, 2)

PSij(1, 3)

PSij(1, 4)

STapprox

STglobal

SRij(2, 2)

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0
2	0	0	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0
3	0	0	0	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0
4	0	0	0	0	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0
5	0	0	0	0	0	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0
6	0	0	0	0	0	0	0.6589	0.6589	0.6589	0.6589	0.6589	0.6589	0
7	0	0	0	0	0	0	0	0.6589	0.6589	0.6589	0.6589	0.6589	0
8	0	0	0	0	0	0	0	0	0.6589	0.6589	0.6589	0.6589	0
9	0	0	0	0	0	0	0	0	0	0.6589	0.6589	0.6589	0
10	0	0	0	0	0	0	0	0	0	0	0.6589	0.6589	0
11	0	0	0	0	0	0	0	0	0	0	0	0.6589	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0.6589
13	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0

Activate Windows

Gambar 7.8 partition score Srij

	SR	SR_sum	SRI	SRij	SRij(2, 2)	PSi	PSij	PSij(1, 2)	PSij(1, 3)	PSij(1, 4)	STapprox	STglobal	
	PSij(1, 2)												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	0.7052	1	1	1	1	0.7052	0.7052	1	1	1	0
2	0	0	0.7052	1	1	1	1	0.7052	0.7052	1	1	1	0
3	0	0	0	0.7052	0.7052	0.7052	0.7052	1	1	0.7052	0.7052	0.7052	0
4	0	0	0	0	1	1	1	0.7052	0.7052	1	1	1	0
5	0	0	0	0	0	1	1	0.7052	0.7052	1	1	1	0
6	0	0	0	0	0	0	1	0.7052	0.7052	1	1	1	0
7	0	0	0	0	0	0	0	0.7052	0.7052	1	1	1	0
8	0	0	0	0	0	0	0	0	1	0.7052	0.7052	0.7052	0
9	0	0	0	0	0	0	0	0	0	0.7052	0.7052	0.7052	0
10	0	0	0	0	0	0	0	0	0	0	1	1	0
11	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 7.9 pairwise support PSij

+2 STapprox Pr{1, 2}{1, 1} Pr{1, 2}{1, 2}

101x4 double

	1	2	3	4
1	0	0.9537	0.8115	0.9909
2	0	0.9451	0.8016	0.9832
3	0	0.4843	0.9767	0.8091
4	0	0.9299	0.7818	0.9611
5	0	0.9213	0.7720	0.9535
6	0	0.9113	0.7620	0.9436
7	0	0.9008	0.7519	0.9322
8	0	0.4577	0.9247	0.7576
9	0	0.4483	0.9173	0.7497
10	0	0.8810	0.7217	0.8989
11	0	0.8736	0.7126	0.8941
12	0	0.8610	0.8197	0.6709
13	0	0.4262	0.8776	0.7195
14	0	0.8260	0.8604	0.7069
15	0	0.8356	0.7328	0.8837
16	0	0.8247	0.7335	0.8817
17	0	0.8181	0.7705	0.6423
18	0	0.8100	0.6596	0.8248
19	0	0.3959	0.8170	0.6671
20	0	0.7944	0.3265	0.7030
21	0	0.7835	0.7393	0.6139

Gambar 7.10 approximate variant  $ST_{approx}(i,K)$

+2 STapprox STglobal Pr{1, 2}{1, 1}

1x4 double

	1	2	3	4
1	0	0.4625	0.4440	0.4580

Gambar 7.11 global stability index  $ST_{global}$