

Computing distance-regular graph and association scheme parameters in SageMath with `sage-drg`

Janoš Vidali

University of Ljubljana

Live slides on Binder

<https://github.com/janos/sage-drg>

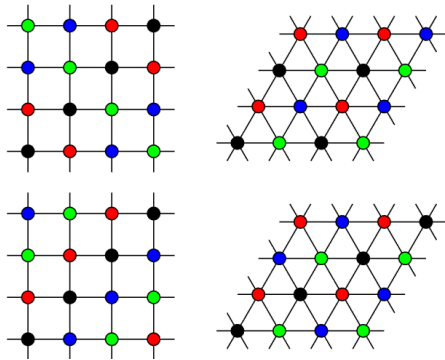


Association schemes

- Association schemes were defined by Bose and Shimamoto in 1952 as a theory underlying experimental design.
- They provide a unified approach to many topics, such as
 - combinatorial designs,
 - coding theory,
 - generalizing groups, and
 - strongly regular and distance-regular graphs.

Examples

- Hamming schemes: $X = \mathbb{Z}_n^d, x R_i y \Leftrightarrow \text{weight}(x - y) = i$
- Johnson schemes: $X = \{S \subseteq \mathbb{Z}_n \mid |S| = d\} (2d \leq n),$
 $x R_i y \Leftrightarrow |x \cap y| = d - i$



Definition

- Let X be a set of vertices and $\mathcal{R} = \{R_0 = \text{id}_X, R_1, \dots, R_D\}$ a set of symmetric relations partitioning X^2 .
- (X, \mathcal{R}) is said to be a D -class association scheme if there exist numbers $p_{ij}^h (0 \leq h, i, j \leq D)$ such that, for any $x, y \in X,$
 $x R_h y \Rightarrow |\{z \in X \mid x R_i z R_j y\}| = p_{ij}^h$
- We call the numbers $p_{ij}^h (0 \leq h, i, j \leq D)$ intersection numbers.

Main problem

- Does an association scheme with given parameters exist?
 - If so, is it unique?
 - Can we determine all such schemes?
- Lists of feasible parameter sets have been compiled for strongly regular and distance-regular graphs.
- Recently, lists have also been compiled for some Q -polynomial association schemes.
- Computer software allows us to efficiently compute parameters and check for existence conditions, and also to obtain new information which would be helpful in the construction of new examples.

Bose-Mesner algebra

- Let A_i be the binary matrix corresponding to the relation $R_i (0 \leq i \leq D)$.
- The vector space \mathcal{M} over \mathbb{R} spanned by $A_i (0 \leq i \leq D)$ is called the Bose-Mesner algebra.
- \mathcal{M} has a second basis $\{E_0, E_1, \dots, E_D\}$ consisting of projectors to the common eigenspaces of $A_i (0 \leq i \leq D)$.
- There are nonnegative constants q_{ij}^h , called Krein parameters, such that

$$E_i \circ E_j = \frac{1}{|X|} \sum_{h=0}^d q_{ij}^h E_h,$$

where \circ is the entrywise matrix product.

Parameter computation: general association schemes

```
In [2]: import drg
p = [[[1, 0, 0, 0], [0, 6, 0, 0], [0, 0, 3, 0], [0, 0, 0, 6]],
      [[0, 1, 0, 0], [1, 2, 1, 2], [0, 1, 0, 2], [0, 2, 2, 2]],
      [[0, 0, 1, 0], [0, 2, 0, 4], [1, 0, 2, 0], [0, 4, 0, 2]],
      [[0, 0, 0, 1], [0, 2, 2, 2], [0, 2, 0, 1], [1, 2, 1, 2]]]
scheme = drg.ASParameters(p)
scheme.kreinParameters()
```

```
Out[2]: 0: [1 0 0 0]
        [0 6 0 0]
        [0 0 3 0]
        [0 0 0 6]

1: [0 1 0 0]
    [1 2 1 2]
    [0 1 0 2]
    [0 2 2 2]

2: [0 0 1 0]
    [0 2 0 4]
    [1 0 2 0]
    [0 4 0 2]

3: [0 0 0 1]
    [0 2 2 2]
    [0 2 0 1]
    [1 2 1 2]
```

Metric and cometric schemes

- If $p_{ij}^h \neq 0$ (resp. $q_{ij}^h \neq 0$) implies $|i - j| \leq h \leq i + j$, then the association scheme is said to be **metric** (resp. **cometric**).
- The parameters of a **metric** association scheme can be determined from the **intersection array** $\{b_0, b_1, \dots, b_{D-1}; c_1, c_2, \dots, c_D\}$ ($b_i = p_{1,i+1}^i, c_i = p_{1,i-1}^i$).
- The parameters of a **cometric** association scheme can be determined from the **Krein array** $\{b_0^*, b_1^*, \dots, b_{D-1}^*; c_1^*, c_2^*, \dots, c_D^*\}$ ($b_i^* = q_{1,i+1}^i, c_i^* = q_{1,i-1}^i$).
- **Metric** association schemes correspond to **distance-regular graphs**.

Parameter computation: metric and cometric schemes

```
In [3]: from drg import DRGParameters
syl = DRGParameters([5, 4, 2], [1, 1, 4])
syl
```

```
Out[3]: Parameters of a distance-regular graph with intersection array {5, 4, 2; 1, 1, 4}
```

```
In [4]: syl.order()
```

```
Out[4]: 36
```

```
In [5]: from drg import QPolyParameters
q225 = QPolyParameters([24, 20, 36/11], [1, 30/11, 24])
q225
```

```
Out[5]: Parameters of a Q-polynomial association scheme with Krein array {24, 20, 36/11; 1, 1, 30/11, 24}
```

```
In [6]: q225.order()
```

```
Out[6]: 225
```

```
In [7]: syl.pTable()
```

```
Out[7]: 0: [ 1 0 0 0]
        [ 0 5 0 0]
        [ 0 0 2 0]
        [ 0 0 0 10]

1: [0 1 0 0]
   [1 0 4 0]
   [0 4 8 8]
   [0 0 8 2]

2: [ 0 0 1 0]
   [ 0 1 2 2]
   [ 1 2 11 6]
   [ 0 2 6 2]

3: [ 0 0 0 1]
   [ 0 0 4 1]
   [ 0 4 12 4]
   [ 1 1 4 4]
```

```
In [8]: syl.kreinParameters()
```

```
Out[8]: 0: [ 1 0 0 0]
        [ 0 16 0 0]
        [ 0 0 10 0]
        [ 0 0 0 9]

1: [ 0 1 0 0]
   [ 1 44/5 22/5 9/5]
   [ 0 22/5 2 18/5]
   [ 0 9/5 18/5 18/5]

2: [ 0 0 1 0]
   [ 0 176/25 16/5 144/25]
   [ 1 16/5 4 9/5]
   [ 0 144/25 9/5 36/25]

3: [ 0 0 0 1]
   [ 0 16/5 32/5 32/5]
   [ 0 32/5 2 8/5]
   [ 1 32/5 8/5 0]
```

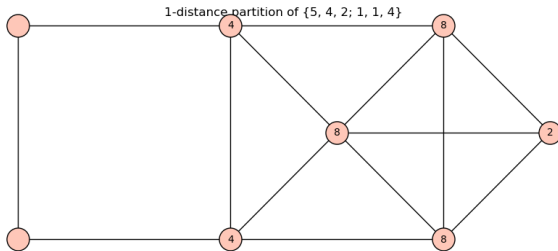
```
In [9]: syl.distancePartition()
```

```
Out[9]: Distance partition of {5, 4, 2; 1, 1, 4}
```



```
In [10]: syl.distancePartition(1)
```

```
Out[10]:
```



Feasibility checking

A parameter set is called **feasible** if it passes all known **existence conditions**.

Let us verify that $H(3, 3)$ is feasible.

```
In [14]: f1.check_feasible()
```

No error has occurred, since all existence conditions are met.

Triple intersection numbers

- In some cases, **triple intersection numbers** can be computed.
- **Nonexistence** of some Q -polynomial association schemes has been proven by obtaining a **contradiction** in **double counting** with triple intersection numbers.

```
In [16]: q225.check_quadruples()
```

```
-----
InfeasibleError Traceback (most recent call last)
<ipython-input-16-40f750f5d8a3> in <module>()
----> 1 q225.check_quadruples()

/home/janos/repos/git/sage-drg/jupyter/2019-07-04-fpsac/drg/assoc_scheme.py in
check_quadruples(self, solver)
    685         "d(w, y) = %d, d(w, z) = %d, "
    686         "d(x, y) = %d, d(x, z) = %d, "
--> 687         "d(y, z) = %d" % (sd + st))
    688         if len(r[st]) < 1:
    689             zero[st] = {(sh, si, sj)}

InfeasibleError: found forbidden quadruple wxyz with d(w, x) = 1, d(w, y) = 1,
d(w, z) = 1, d(x, y) = 3, d(x, z) = 3, d(y, z) = 3
```

Integer linear programming has been used to find solutions to multiple systems of **linear Diophantine equations**, **eliminating** inconsistent solutions.

Parameter computation: parameters with variables

Let us define a **one-parametric family** of **intersection arrays**.

```
In [11]: r = var("r")
f = DRGParameters([2*r^2*(2*r+1), (2*r-1)*(2*r^2+r+1), 2*r^2], [1, 2*r^2, r*(4*
r^2-1)])
f
```

```
Out[11]: Parameters of a distance-regular graph with intersection array {4*r^3 + 2*r^2,
4*r^3 + r - 1, 2*r^2; 1, 2*r^2, 4*r^3 - r}
```

```
In [12]: f1 = f.subs(r == 1)
f1
```

```
Out[12]: Parameters of a distance-regular graph with intersection array {6, 4, 2; 1, 2,
3}
```

The parameters of f_1 are known to **uniquely determine** the **Hamming scheme** $H(3, 3)$.

```
In [13]: f2 = f.subs(r == 2)
f2
```

```
Out[13]: Parameters of a distance-regular graph with intersection array {40, 33, 8; 1,
8, 30}
```

Let us now check whether the second member of the family is feasible.

```
In [15]: f2.check_feasible()
```

```
-----
InfeasibleError Traceback (most recent call last)
<ipython-input-15-83a4aafdb73c> in <module>()
----> 1 f2.check_feasible()

/home/janos/repos/git/sage-drg/jupyter/2019-07-04-fpsac/drg/drg.py in check_f
easible(self, checked, skip, derived)
    682         for name, check in checks:
    683             if name not in skip:
--> 684                 check()
    685             if not derived:
    686                 return

/home/janos/repos/git/sage-drg/jupyter/2019-07-04-fpsac/drg/drg.py in check_f
amily(self)
    643         in zip(self.b[:-1] + self.c[1:], b + c)], v
ars)
    644         if any(checkConditions(cond, sol) for sol in sols):
--> 645             raise InfeasibleError(refs = ref)
    646
    647     def check_feasible(self, checked = None, skip = None, derived = Tr
ue):

InfeasibleError: nonexistence by JurišićVidalil12
```

In this case, **nonexistence** has been shown by **matching** the parameters against a list of **nonexistent families**.