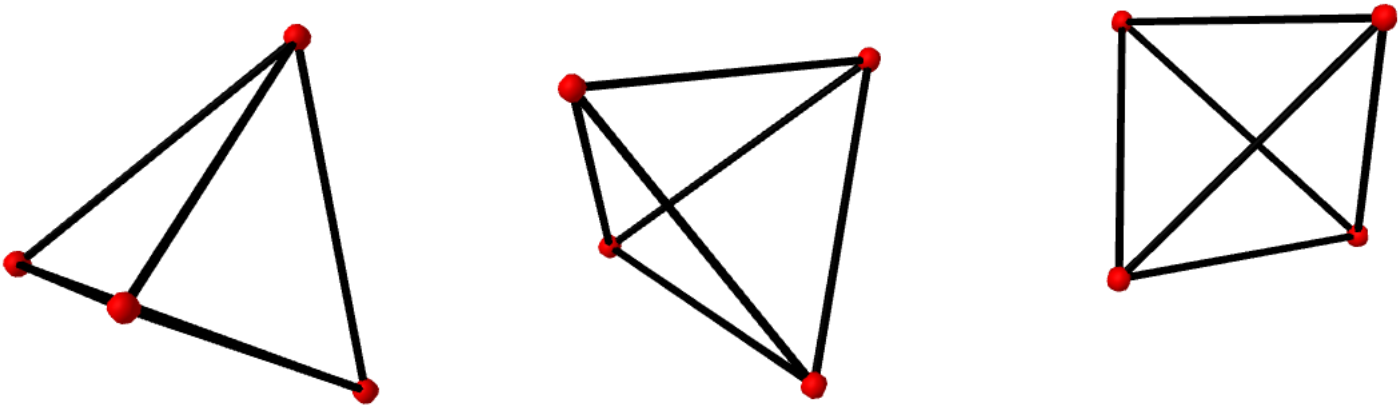
EKR Documentation
2021

Cody Antal



Contents

Part I

Usage

1 Installation

Assuming working SageMath and Gurobi installations

1. Copy all files (except perhaps the data folder) into a common directory
2. Using the Sage shell, navigate to this directory and attach the attach.sage file

   ```
   attach("attach.sage")
   ```

From the Sage shell you now have access to all the needed code. It may also be beneficial to look at the _solve_lp() function in the "EKR Determiner.sage" file. It calls the command needed to run Gurobi. The exact command needed may change based on your system, so if any errors with gurobi occur, this is the place to check.

2 Generate Data

To generate data files for a set of groups:

1. In the installation folder, create a folder called Data.
2. In the Data folder, create folders for each degree you wish to test e.g If you are testing groups of degree 12 make sure that Data/12 exists.
3. Using the Sage shell fill a list with all the groups you want to test

   ```
   groups_to_test = [TransitiveGroup(12, 4), TransitiveGroup(16, 435), ...]
   ```

   This can be done manually like the above, or if you wish to test all groups of some degree and some maximum order

   ```
   groups_to_test = [G for G in TransitiveGroups(degree) if G.order() <= max_order]
   ```

4. Once you have the list, to generate the data simply:

   ```
   Data_Generator(groups_to_test)
   ```

   This will automatically begin the computation and save the results.

5. It is not uncommon for GAP (an included library of Sage) to raise an error complaining about a memory limit. To get around this, the program will automatically ignore the error and write the group it occured on to the skipped.txt file. Check this file after the computation finishes to see if any groups were effected. Then you can redo the computation for just those groups.

3 Program Structure

The basic flow of the program is

```
compute_basic_information_about_group() # The Common class
compute_the_ekr_status_of_the_group() # The EKR_Determiner class
compute_the_ekrm_status_of_the_group() # The EKRM_Determiner class
compute_the_strict_ekr_status_of_the_group() # The Strict_EKR_Determiner class
save_the_data() # The Data_Generator class
```

the exact details of each class are described in the next part

Part II

Reference

4 Common Class

4.1 Purpose

The common class pre-computes a set of frequeuntly used information e.g eigenvalues, subgroups, cocliques. An object of this type will always be named either common or, much more frequently, G in the source code. The constructor takes the form:

```
Common(group)
```

Where group is a Sage (Transitive) Permutation group

4.2 Attributes

Attribute	Value
group	Underlying sage transitive group
degree	degree of group
order	order of group
transitivity	transitivity of group
identity	identity element of group
characters	irreducible characters of group
conjugacy_classes	conjugacy classes of group
derangement_classes	conjugacy classes of derangements of group
subgroups	conjugacy classes of subgroups of group
eigenvalues	eigenvalues of group's derangement graph
eigenvalues_with_multiplicities	eigenvalues but with repeated eigenalues included
max_eigenvalue	maximum eigenvalue
min_eigenvalue	minimum eigenvalue
n_cliques	elements of subgroups of size degree that form cliques
stabilizer_sized_cocliques	elements of subgroups of size $\frac{order}{degree}$ that form cocliques
larger_than_stabilizer_cocliques	elements of subgroups of size larger than $\frac{order}{degree}$ that form cocliques

4.3 Methods

_get_derangement_classes

Input: None
Output: List of conjugacy classes of derangements

_get_eigenvalues

Input: None
Output: tuple whose first component is the eigenvalues of the group without included multiplicities, and whose second argument is the eigenvalues with included multiplicities.

_get_n_cliques

Input: None
Output: List of (conjugacy class representative) subgroups that form cliques of size n

_get_stabilizer_sized_cocliques

Input: None
Output: List of (conjugacy class representative) subgroups that form cocliques of size $\frac{|G|}{n}$

_get_larger_than_stabilizer_cocliques

Input: None
Output: List of (conjugacy class representative) subgroups that form cocliques of size larger than $\frac{|G|}{n}$

_get_eigenvalues_subgroup

Input: subgroup of group
Output: Eigenvalues of adjacency matrix of subgroup without multiplicities

5 EKR Determiner Class

5.1 Purpose

The EKR Determiner class determines if the group has the EKR property and reasons for how we know. The constructor takes the form:

```
EKR_Determiner(common)
```

where *common* is an object of the Common class

5.2 Attributes

Attribute	Value
G	common object (i.e object with useful precomputed values)
has_ekr	boolean or None (None if we can not tell if it is EKR or not)
reasons	an array of strings describing how we know the value of has_ekr
weightings	If a weighting on derangement classes gives EKR, this is an array of those weightings.

5.3 Methods

_ratiobound_gives_ekr

Input: None
Output: boolean describing if we get EKR immediatly from the ratiobound

_weighting_gives_ekr

Input: None
Output: boolean describing if a weighting gives EKR

_get_coefficents

Input: None
Output: An array of arrays. Each element of the big array is an array of the coefficents needed for linear programming i.e if 5 x0 + 2 x1 - 3.14 x2 is a constraint for the linear program, then [5, 2, -3.14] is one of the elements

_get_open_conjugacy_classes

Input: None
Output: An array of 2-tuples. Each tuple gives two distinct conjugacy classes C_1 and C_2 that are related by $c_1^{-1} \in C_2$ for all $c_1 \in C_1$ (and the reverse). The values are needed as the related conjugacy classes must have the same weightings if the linear program is to be useful.

_create_lp

Input: coefficents and open_conjugacy_classes. The coefficents are expected to be the output of _get_coefficents, and the open_conjugacy_classes the output of the above
Output: Creates a filled out temp.lp in the gurobi folder

_create_linear_combination_string

Input: coefficents and bound. coefficents must be a list of numbers, and bound a string
Output: A string taking the form "coefficent[0] x0 + ... + coefficents[last] xlast" + bound. The bound tends to look like " ¿= -1" as we only use this function for generating the lp file

_solve_lp

Input: None
Output: Creates temp.sol in the gurobi folder. The function may need to be rewritten in order to work for your particular gurobi installation/operating system.

_read_sol

Input: None
Output: A 2-tuple. First argument is a boolean describing if the weightings found by gurobi are good enough for EKR. The second are the weightings (None if the weightings do not work).

6 EKRM Determiner Class

6.1 Purpose

The EKRM Determiner class determines if the group has the EKRM property and reasons for how we know. The constructor takes the form:

```
EKRM_Determiner(common, ekr_determiner)
```

6.2 Attributes

Attribute	Value
G	Common object
ekr_determiner	EKR_Determiner object
has_ekrm	boolean or None
reasoms	Reasons for EKRM classification

6.3 Methods

_cliques_give_ekrm

Input: decomposition of permutation representation i.e array of coefficents for each irreducible character in the permutation representation
Output: boolean describing if we can prove the group has the EKRM property using cliques of max size

_cocliques_disprove_ekrm

Input: decomposition Output: boolean describing if we can disprove that the group has the EKRM property using cocliques of max size.

_weighted_eigenvalues_give_ekrm

Input: decomposition
Output: boolean describing if we can prove that the group has the EKRM property using the weighted eigenvalues

_get_permutation_decomposition

Input: None
Output: Array of the coefficents for each irreducible character in the permutation representation

_get_characters_not_in_decomposition

Input: decomposition
Output: All irreducible characters with coefficent 0 in the decomposition

_get_characters_in_decomposition

Input: decomposition
Output: All irreducible characters with non-zero coefficent in the decomposition

_get_character_sum

Input: character, collection. character is an irreducible character and collection is a list of group elements
Output: $\sum_{g \in \text{collection}} \chi(g)$

7 Strict EKR Determiner Class

7.1 Purpose

The Strict EKR Determiner class determines if the group has the strict EKR property and reasons for how we know. The constructor takes the form:

```
Strict_EKR_Determiner(common, ekr_determiner, ekrm_determiner)
```

7.2 Attributes

Attribute	Value
G	common object
has_strict_ekr	boolean or None
reasons	list of reasons we know has_strict_ekr

7.3 Methods

_module_method_gives_sekr

Input: None
Output: boolean describing if the module method gives strict EKR (for two transitive groups)

_coclique_disproves_sekr

Input: None
Output: boolean describing if we have a coclique that disproves strict EKR for this group

_get_m_matrix

Input: None
Output: numpy array representing the large matrix in the module method

_get_n_matrix

Input: m_matrix
Output: numpy array representing the smaller matrix in the module method

_get_matrix_rank

Input: matrix
Output: rank of matrix

_collection_is_the_stabilizer_of_a_point

Input: collection of group elements
Output: boolean describing if the collection is the stabilizer of some point

8 Data Generator Class

8.1 Purpose

Manages the flow of all computations needed to generate data for each group, then handles saving the data to the correct file. Construct takes the form:

```
Data_Generator(groups)
```

Where groups is a list of Sage Transitive Permutation groups. Calling the constructor will begin the computation and save all data.

8.2 Methods

_get_number

Input: common object
Output: The "number" in the groups name e.g Transitive Group number "number" of degree 12

_get_nice_eigenvalues

Input: commonn object
Output: The eigenvalues in the form [(eigenvalue, multiplicity), ...]

_save

Input: data (dictionary defined in class) Output: Saves data to ./degree/number.txt