images

Minimal and Canonical images

1.3.3

27 August 2024

Christopher Jefferson

Markus Pfeiffer

Rebecca Waldecker

Eliza Jonauskyte

Christopher Jefferson
Email: caj21@st-andrews.ac.uk
Homepage: http://caj.host.cs.st-andrews.ac.uk/

Markus Pfeiffer
Email: markus.pfeiffer@morphism.de
Homepage: http://www.morphism.de/~markusp/

Rebecca Waldecker
Email: rebecca.waldecker@mathematik.uni-halle.de
Homepage: http://conway1.mathematik.uni-halle.de/~waldecker/

Eliza Jonauskyte
Email: ej31@st-andrews.ac.uk

Copyright

Contents

Chapter 1

The Images Package

The GAP package Images implements efficient algorithms to calculate minimal images and canonical images of several different types of objects ,including sets and sets of sets, in permutation groups.

If you want help on applying Images to your problem, please contact the authors, who will be happy to look into if your problem can be solved with Images.

Chapter 2

Minimal and Canonical Images

Given a group G and action A, the minimal image of an object O is the smallest image of O under any element of G, under the action A.

As a more concrete example, let us consider the minimal image of the set [2,3,5,7] under a group G.

We can calculate all the images of our set under G, then choose the smallest one.

Example

```
gap> G := Group((1,2,3)(4,5,6)(7,8,9),(1,4,7)(2,5,8)(3,6,9));;
gap> List(G, g -> OnSets([2,3,5,7], g) );
[ [ 2, 3, 5, 7 ], [ 1, 2, 4, 9 ], [ 1, 3, 6, 8 ], [ 2, 4, 8, 9 ],
  [ 1, 6, 7, 8 ], [ 3, 5, 7, 9 ], [ 1, 5, 6, 8 ], [ 3, 4, 5, 7 ],
  [ 2, 4, 6, 9 ] ]
gap> Minimum(List(G, g -> OnSets([2,3,5,7], g) ) );
[ 1, 2, 4, 9 ]
```

This is very inefficient, as it requires enumerating all members of G. The images package produces a function MinimalImage, which performs this same operation more efficiently.

Example

```
gap> LoadPackage("images", false);
true
gap> MinimalImage(G, [2,3,5,7], OnSets);
[ 1, 2, 4, 9 ]
```

The most common use of MinimalImage is to categorise objects into equivalence classes. This next example shows [2,3,5,7] and [1,6,7,8] are in the same orbit, while [3,5,7,8] is in a different orbit.

Example

```
gap> MinimalImage(G, [2,3,5,7], OnSets);
[ 1, 2, 4, 9 ]
gap> MinimalImage(G, [1,6,7,8], OnSets);
[ 1, 2, 4, 9 ]
gap> MinimalImage(G, [3,5,7,8], OnSets);
[ 1, 2, 6, 8 ]
```

In this situation, we do not really need the minimal image, just a method of telling if two sets are in the same equivalence class.

Motivated by this, this package provides CanonicalImage. CanonicalImage(G,O,A) returns some image of O by an element of G under the action A, guaranteeing that if two objects O1 and O2

are in the same orbit of G then CanonicalImage(G,O1,A) = CanonicalImage(G,O2,A). However, the canonical image is not "minimal" under any sensible ordering. The advantage of CanonicalImage is that it is much faster than MinimalImage, often by orders of magnitude.

WARNING: The value of MinimalImage will remain identical between versions of GAP and the Images package, unless bugs are discovered. This is NOT true for CanonicalImage.

2.1 Function documentation

2.1.1 MinimalImage

▷ `MinimalImage(G, pnt[, act][, Config])` (function)
▷ `IsMinimalImage(G, pnt[, act][, Config])` (function)
▷ `MinimalImagePerm(G, pnt[, act][, Config])` (function)

`MinimalImage` returns the minimal image of *pnt* under the group *G*. `IsMinimalImage` returns a boolean which is `true` if `MinimalImage` would return *pnt* (so the value is it's own minimal image).

`MinimalImagePerm` returns the permutation which maps *pnt* to its minimal image.

The option *Config* defines a number of advanced configuration options, which are described in 'ImagesAdvancedConfig'.

2.1.2 IsMinimalImageLessThan

▷ `IsMinimalImageLessThan(G, A, B[, act][, config])` (function)

`IsMinimalImageLessThan` checks if the minimal image of *A* under the group *G* is smaller than *B*.

It returns MinImage.Smaller, MinImage.Equal or MinImage.Larger, if the minimal image of *A* is smaller, equal or larger than *B*.

The option *Config* defines a number of advanced configuration options, which are described in 'ImagesAdvancedConfig'.

2.1.3 CanonicalImage

▷ `CanonicalImage(G[, pnt][, act][, Config])` (function)
▷ `IsCanonicalImage(G[, pnt][, act][, Config])` (function)
▷ `CanonicalImagePerm(G[, pnt][, act][, Config])` (function)

`CanonicalImage` returns a canonical image of *pnt* under the group *G*. `IsCanonicalImage` returns a boolean which is `true` if `CanonicalImage` would return *pnt* (so the value is it's own minimal image).

`CanonicalImagePerm` returns the permutation which maps *pnt* to its minimal image.

By default, these functions use the fastest algorithm for calculating canonical images, which is often changed in new versions of the package. The option *Config* defines a number of advanced configuration options, which are described in 'ImagesAdvancedConfig'. These include the ability to choose the canonicalising algorithm used.

2.1.4 ImagesAdvancedConfig

▷ `ImagesAdvancedConfig` (global variable)

This section describes the advanced configuration options for both `MinimalImage` (2.1.1) and `CanonicalImage` (2.1.3). Assume we have called `MinimalImage` (2.1.1) or `CanonicalImage` (2.1.3) with arguments `(G,O,A)`.

`order`
: The search ordering used while building the image. There are many configuration options available. We shall list here just the three most useful ones. A full list is in the paper "Minimal and Canonical Images" by the authors of this package.

  `CanonicalConfig_Minimum`
  : Lexicographically smallest set -- same as using MinimalImage.

  `CanonicalConfig_FixedMinOrbit`
  : Lexicographically smallest set under the ordering of the integers given by the MinOrbitPerm function.

  `CanonicalConfig_RareRatioOrbitFixPlusMin`
  : The current best algorithm (default)

`stabilizer`
: The group `Stabilizer(G,O,A)`, or a subgroup of this group; see `Stabilizer` (**Reference: Stabilizer**). If this group is large, it is more efficient to pre-calculate it. Default behaviour is to calculate the group, pass `Group(())` to disable this behaviour. This is not checked, and passing an incorrect group will produce incorrect answers.

`disableStabilizerCheck` **(default** `false`**)**
: By default, during search we perform cheap checks to try to find extra elements of the stabilizer. Pass true to disable this check, this will make the algorithm MUCH slower if the stabilizer argument is a subgroup.

`getStab` **(default** `false`**)**
: Return the calculated value of `Stabilizer(G,O,A)`. This may return a subgroup rather than the whole stabilizer.

Chapter 3

Installing and Loading the Images Package

To install the Images package simply place Images into the pkg directory of your GAP installation.

3.1 Loading the Images Package

To use the Images Package you have to request it explicitly. This is done by calling `LoadPackage` (**Reference: LoadPackage**):

Example

```
gap> LoadPackage("images");
true
```

Index