MapClass

May 2011

Adam James
Kay Magaard
Sergey Shpectorov
Helmut Volklein

Adam James — Email: jamesa@maths.bham.ac .uk

Kay Magaard — Email: K.Magaard@maths.bham .ac.uk

Sergey Shpectorov — Email: s.shpectorov@bham.ac .uk

Helmut Volklein — Email: voelkle@iem.uni-due. de

Copyright

© 2004-2011 by Adam James, Kay Magaard Sergey Shpectorov, and Helmut Volklein

We adopt the copyright regulations of GAP as detailed in the copyright notice in the GAP manual.

Contents

1 Introduction and Main Functions **4**
- 1.1 Background Material . . . 4
- 1.2 Overview of Main Functions . . . 4
  - 1.2.1 AllMCOrbits . . . 4
  - 1.2.2 AllMCOrbitsCore . . . 5
  - 1.2.3 GeneratingMCOrbits . . . 5
  - 1.2.4 GeneratingMCOrbitsCore . . . 5
  - 1.2.5 MappingClassOrbit . . . 6
  - 1.2.6 PrepareMinTree . . . 6
  - 1.2.7 MinimizeTuple . . . 6
  - 1.2.8 EasyMinimizeTuple . . . 6
  - 1.2.9 IsInOrbit . . . 6
  - 1.2.10 FindTupleInOrbit . . . 6
  - 1.2.11 IsEqualOrbit . . . 7
  - 1.2.12 SelectTuple . . . 7
  - 1.2.13 NumberGeneratingTuples . . . 7
  - 1.2.14 TotalNumberTuples . . . 7
  - 1.2.15 CalculateTuplePartition . . . 7
  - 1.2.16 RestoreOrbitFromFile . . . 7
  - 1.2.17 SaveOrbitToFile . . . 7
- 1.3 Key Data Structures . . . 7
  - 1.3.1 The Orbit Record . . . 8
  - 1.3.2 The Tuple Table . . . 8
- 1.4 A Sample Session . . . 9

A Installation **12**

Chapter 1

Introduction and Main Functions

This chapter provides an overview of the background material, and provides documentation for the main functions and data structures of the MAPCLASS package.

1.1 Background Material

Let G be a finite group, and let $\sigma = (\sigma_1, ..., \sigma_{2g+r})$ (for natural numbers g and r) be a tuple of elements of G satisifying the relation

$$\prod_{i=1}^{g}[\sigma_i, \sigma_{g+i}]\prod_{i=1}^{r}\sigma_{2g+i} = 1$$

If we associate the elements of the tuple σ with the standard generators of the fundamental group of a compact connected surface (genus g, r punctures), then the mapping class group acts on the fundamental group generators inducing an action on the set of tuples. The *mapping class orbit* is then the orbit under this action. Note that we take these orbits up to conjugation. Often we are only concerned with tuples which generate the group G, such tuples are said to be *generating*

The package can be used to compute the mapping class orbits for given G and a set of conjugacy classes $(C_1, ..., C_r)$ (although the programs expect a tuple of class representatives). We call the tuple $(g; C_1, ..., C_r)$ the signature. The package is an extension of the *Braid* package for GAP

1.2 Overview of Main Functions

The following are the principal ways for calculating the mapping class orbits for a given signature and group.

1.2.1 AllMCOrbits

◊ `AllMCOrbits(group, genus, tuple)` (function)

This function calculates the orbit for the given group, genus and tuple. This function is a wrapper for the function `AllMCOrbitsCore` (1.2.2), and so can make use of GAP's `OptionsStack`. The options are described in more detail in the documentation for `AllMCOrbitsCore` (1.2.2)

1.2.2 AllMCOrbitsCore

◊ AllMCOrbitsCore(*group, genus, tuple, partition, constant*) (function)

This function calculates the orbit for the given group, genus and tuple, with the r branch points partitioned as in *partition*. It uses the given *constant* to determine how many of the tuples it want to account for before exiting. This function also make use of GAP's OptionsStack if one desires to alter the algorithm runs. The following options and their defaults are given below.

Option Name	Default Value
InitialSizeCutOff	128
MaximumWeight	40
MinimumWeight	20
InitialWeight	20
BumpUp	7
KnockDown	7
InitialNumberOfRandomTuples	20
InitialCutOffValue	0
HashLength	5000
SaveOrbit	False

The majority of the options are uninteresting and only subtly alter the running of the routine.The options InitialSizeCutOff, MaximumWeight, MinimumWeight, InitialWeight,BumpUp, KnockDown, are the options controlling how the code handles the subgroup weighting discussed in the algorithm overview. The option InitialNumberOfRandomTuples decides how many tuples the routine collects before trying to see if they are in some pre-existing orbit. The option InitialCutOffValue decides at what point we stop searching for new orbits - if only large orbits are of interest then this can be set larger to ignore smaller orbits. Finally the option SaveOrbit which is by default False can be set to the name of a directory in which you want to save orbits. This option then saves the orbits to files in the folder with "_name". So for example if you wish to save your orbits into the file _example then you would run AllMCOrbits(group, genus, tuple: SaveOrbit:="example");. The orbits are then saved in orbits which are named numerically. Following on from the above example then the first orbit will be saved as "example/0".

1.2.3 GeneratingMCOrbits

◊ GeneratingMCOrbits(*group, genus, tuple*) (function)

This calculates the orbits for the given arguments. Unlike the AllMCOrbits (1.2.1) function, GeneratingMCOrbits calculates only those orbits whose tuples generate the whole of our original group.

1.2.4 GeneratingMCOrbitsCore

◊ GeneratingMCOrbitsCore(*group, genus, tuple, partition, constant*) (function)

This calculates the orbits for the given arguments. Unlike the AllMCOrbits (1.2.1) function, GeneratingMCOrbitsCo re calculates only those orb its whose tuples generate the whole of our original group. As with AllMCOrbitsCore (1.2.2), the argument *partition* must be a partition of the conjugacy classes represented in list form. We also have access to the full value of the options stack as in AllMCOrbitsCore (1.2.2).

1.2.5 MappingClassOrbit

◊ MappingClassOrbi t(*group, genus, pri ncipaltuple, partition, tuple*) (function)

Returns: an orbit record for an orbit containing tuple or returns fail

Calculates the orbit of the *tuple* with respect to the given *group*, *principaltuple* and *genus*.

1.2.6 PrepareMinTree

◊ PrepareMinTree(*principaltuple, g roup, ourR, genus*) (function)

Returns: a record with two keys MinimizationTree and MinimumSet. If record is the returned record then record.Minimization Tree is the list encoding the t ree used to help minimize tuples.The list record.MinimumSet is a list of minimal elem ents which is also used to speed up tuple minimization.

1.2.7 MinimizeTuple

◊ MinimizeTuple(*tuple, minimizati onTree, minimumSet, numberOfGenerators*) (function)

Returns: the minimal tuple in the same orbit of *tuple*.

Take the minimisation data provided by PrepareMinTree (1.2.6) and minimizes the given *tuple*.

1.2.8 EasyMinimizeTuple

◊ EasyMinimizeTuple(*group, genus, tup le*) (function)

Returns: the minimal tuple in the same orbit as *tuple*.

1.2.9 IsInOrbit

◊ IsInOrbit(*orbit, tuple*) (function)

Returns: True if the *tuple* lies in the *orbit*.

1.2.10 FindTupleInOrbit

◊ FindTupleInOrbit(*orbit, tuple*) (function)

Returns: the index of *tuple* in orbit.TupleTable if in the *orbit*. If the tuple is not in *orbit* returns fail.

1.2.11 IsEqualOrbit

◊ IsEqualOrbit(*orbit1, orbit2*) (function)

Returns: true if the two orbits are equa l else returns fail.

1.2.12 SelectTuple

◊ SelectTuple(*orbit, index*) (function)

Returns: the tuple orbit.TupleTable[ind ex].tuple.

1.2.13 NumberGeneratingTuples

◊ NumberGeneratingTupl es(*group, partition, tuple, genus*) (function)

Returns: the total number of possible generating tuples for the group and tuple.

1.2.14 TotalNumberTuples

◊ TotalNumberTuples(*group, tuple, gen us*) (function)

Returns: the total number of tuples we have to account for.

1.2.15 CalculateTuplePartition

◊ CalculateTuplePartit ion(*group, tuple*) (function)

Returns: A partition of $1,\ldots,r$ where r is the length of the tuple.

The function returns a partition of $1,\ldots,r$ such that i and j lie in the same block if and only if the elements tuple[i] and tuple[j] are member of the same conjugacy class. The program currently requires that the elements of the tuple be ordered such that if tuple[i] and tuple[j] are in the same conjugacy class with $i \leq j$ then so istuple[k] for all $i \leq k \leq j$.

1.2.16 RestoreOrbitFromFile

◊ RestoreOrbitFromFile (*n, name[, path]*) (function)

Returns: the n-th orbit record from the project with project "name"

By default the function searches the current working directory for the saved project folder and searches inside this for the n-th orbit. If no such orbit exists it returns fail. If an optional argument *path* is provided then it searches this path for a folder with the name specified (note that path expects a Directory object). If an orbit exists then it is returned as a record as outlined in the data structure section.

1.2.17 SaveOrbitToFile

◊ SaveOrbitToFile(*orbit, n, name*) (function)

Saves the orbit to filename "n" in the directory '_name'. The directory must alrea dy exist.

1.3 Key Data Structures

Many of the above functions require or return key data structures which we aim to document.

1.3.1 The Orbit Record

Many of the functions return or expect an orbit "object". This object is in fact record with the following values:

- `PrincipalFiniteGrou p` - the finite group
- `OurG` - genus
- `OurR` - length of our primary t uple
- `OurN` - number of points on w hich our group acts
- `NumberOfGenerators` - 2 `OurG`+ `OurR`
- `OurFreeGroup` - a free group on `NumberOfGenerators` letters
- `AbsGens` - generators for `OurFreeGroup`
- `OurAlpha` - generators of `OurFreeGroup`corresponding to the α_i type loops in the fundamental group (the first g elements of `AbsGens`)
- `OurBeta` - elements of `OurFreeGroup` corresponding to β type loops
- `OurGamma` - generators of `OurFreeGroup` corresponding to the loo ps around branch points
- `TupleTable` - a table containing all th e tuples in the orbit
- `HashLength`
- `Hash`
- `PrimeCode`
- `OurAction`
- `ActionOnOrbit`
- `MinimizationTree`- minimization structure
- `MinimumSet`- minimizaton structure

1.3.2 The Tuple Table

The tuple table is a list. Each element of the list is a record with the names, tuple and next. If `orbit` is an orbit object then `orbit.TupleTable[n]. tuple` returns the tuple at index n of the tuple table.

1.4 A Sample Session

We demonstate how one might use the package.

Example

```
gap> group:=Alternat    ingGroup(5);
Alt( [ 1 .. 5 ] )
gap> tuple:=[ (1,2)(    3,4), (1,2)(3,4), (1    ,2)(3,4) ]
[ (1,2)(3,4), (1,2)(    3,4), (1,2)(3,4) ]
gap> orbits:=AllMCOr    bits(group,1,tuple);    ;

Total Number of Tupl    es: 189120


Collecting 20 random     tuples... done

Cleaning done; 20 ra    ndom tuples remainin    g


Orbit 1:

Length=3072
Generating Tuple  =[     (1,2,4,5,3), (1,4,5,    2,3), (1,2)(4,5),
(1,4)(2,3), (2,5)(3,    4) ]
Generated subgroup s    ize=60
Centralizer size=1
4800 tuples remainin    g
Cleaning current orb    it...
Cleaning a list of 2    0 tuples
Random Tuples Remain    ing: 0
Cleaning done; 0 ran    dom tuples remaining


Collecting 20 random     tuples... done

Cleaning orbit 1
Cleaning a list of 2    0 tuples
Random Tuples Remain    ing: 0

Cleaning done; 0 ran    dom tuples remaining


Collecting 40 random     tuples... done

Cleaning orbit 1
Cleaning a list of 4    0 tuples
Random Tuples Remain    ing: 3

Cleaning done; 3 ran    dom tuples remaining


Orbit 2:

Length=32
```

```
Generating Tuple  =[     (1,4)(2,3), (1,2)(3,    4), (1,4)(2,3), (1,2    )(3,4),
(1,3)(2,4) ]
Generated subgroup s   ize=4
Centralizer size=4
4320 tuples remainin   g
Cleaning current orb   it...
Cleaning a list of 3    tuples
Random Tuples Remain   ing: 2
Cleaning done; 2 ran   dom tuples remaining


Orbit 3:

Length=72
Generating Tuple  =[     (1,5,2), (1,3,2), (1    ,2)(3,5), (1,3)(2,5)    ,
(1,3)(2,5) ]
Generated subgroup s   ize=12
Centralizer size=1
0 tuples remaining
Cleaning current orb   it...
Cleaning a list of 2    tuples
Random Tuples Remain   ing: 0
Cleaning done; 0 ran   dom tuples remaining

gap> # Check we have    as many orbits as it    says...
gap> Length(orbits);
3
gap> # Inspect the f   irst orbit..
gap> orb1:=orbits[1]   ;;
gap> # How long is o   rb1?
gap> Length(orb1.Tup   leTable);
3072
gap> # Select elemen   t 42 ...
gap> SelectTuple(orb   1, 42);
[ (1,3,4), (1,5,3,2,   4), (1,5)(2,4), (1,2    )(3,5), (2,3)(4,5) ]
gap> # Save the orbi   t to a file...
gap> SaveOrbitToFile   (orb1, 1, "test");
gap> #If the folder    doesn't exist we get    an error..
gap> SaveOrbitToFile   (orb1, 1, "foo");
AppendTo: cannot ope   n '_foo/1' for outpu   t at
CallFuncList( APPEND   _TO, arg );
gap> #
gap> # Now we try fi   nd generating orbits
gap> group:=Symmetri   cGroup(5);
Sym( [ 1 .. 5 ] )
gap> # And we will s   ave them using the `   SaveOrbit` option
gap> GeneratingMCOrb   its(group,1,tuple :    SaveOrbit:="example"    );;

Total Number of Tupl   es: 607680



Collecting 20 genera   ting tuples .. done
```

```
Cleaning done; 20 ra    ndom tuples remainin    g


Orbit 1:
Length=5064
Generating Tuple  =[     (1,3,2,5), (2,4,3),     (1,4)(3,5), (1,3)(2,    5),
(1,4)(3,5) ]
Generated subgroup s    ize=120
Saving orbit to file     _example/0
Centralizer size=1
0 tuples remaining
Cleaning current orb    it...
Cleaning a list of 2    0 tuples
Random Tuples Remain    ing: 0
Cleaning done; 0 ran    dom tuples remaining

gap> generatingorbit    s:=last;;
gap> # How many gene    rating orbits are th    ere?
gap> Length(generati    ngorbits);
1
gap> # Is this orbit     equal to the other o    ne we found earlier
gap> IsEqualOrbit(or    b1, generatingorbits    [1]);
fail
gap> # We can reload     the orbits...
gap> orb2:=RestoreOr    bitFromFile(0, "exam    ple");;
gap> Length(orb2.Tup    leTable);
5064
```

Appendix A

Installation

To Install the package place the ”MapClass” folder into your GAP system’s ’pkg’ directory. If you do not have permission to modify the gap package then the package can be included by appending a local directory to GAP’s root directory using the ’-l’ flag. For more information on GAP’s root directory process try ’?GAP root’ in a GAP session. OR see the online help at http://www.gap-system.org/Manuals/doc/htm/ref/CHAP009.htm#SECT002 To Load the package simply type ’LoadPackage(”mapclass”);’. If the load has been successful the package banner will be shown.

Index

AllMCOrbits, 4
AllMCOrbitsCore, 5

CalculateTuplePartition, 7

EasyMinimizeTuple, 6

FindTupleInOrbit, 6

GeneratingMCOrbits, 5
GeneratingMCOrbitsCore, 5

IsEqualOrbit, 7
IsInOrbit, 6

MappingClassOrbit, 6
MinimizeTuple, 6

NumberGeneratingTuples, 7

Overview, 4

PrepareMinTree, 6

RestoreOrbitFromFile, 7

SaveOrbitToFile, 7
SelectTuple, 7

TotalNumberTuples, 7