CS 299 : Open Source Contribution to GAP

Pranav Joshi (22110197) (email : pranav.joshi@iitgn.ac.in)

Problem Description

Implementing the polynomial time algorithm to find the minimal faithful permutation degree of semi-simple groups as described in the paper *"The Minimal Faithful Permutation Degree of Groups without Abelian Normal Subgroups"* by Dhara Thakkar and Bireswar Das in GAP : Groups Algorithms and Programming, an open source software for computational group theory.

Preparations

I learnt the GAP programming language and wrote the code for the Minimal Generating Set Algorithm using Chief Series to become familiar to the language.

I made a pull request to `gap-system/gap` (main repository of GAP) for the same code. This helped me get used to the file structure of GAP and conventions that the project uses, like XML documentation, tests, linting, and packages. I also found the cause of a bug in SageMath, related to `libgap` (SageMath's interface for GAP) after reading relevant code in the GAP code-base.

To help in understanding the algorithm, I read about these topics:

1. Semi-direct products, Group extensions
2. Core, Orbits, Stabiliser, Derived Series, Subgroup Lattice, Representation
3. Rings, Finite fields, Galois Group, Frobenius Automorphisms

Main Work

Since the paper wasn't available online initially, I talked online with one of the authors, Dhara Thakkar, and got a soft-copy. The paper was mostly mathematical in nature. To understand the algorithm, I had to read about these topics :

1. Classification of Simple Groups
2. Matrix Groups ($GL_n(q), SL_n(q), U_n(q), O_n(q), SU_n(q), SO_n(q), O_n^{\pm}(q), \Omega_n^{\pm}(q), \ldots$) and the corresponding projective groups ($PGL_n(q) \ldots$) .
3. Semilinear transformations and the Groups they form ($P\Gamma L_n(q), P\Sigma L_n(q) \ldots$)

The algorithm for semi-simple groups relies on the existence of a poly-time algorithm for computing the minimal faithful permutation degree of s*imple groups*. Although this algorithm was well known, it was never implemented in GAP till now, So I first implemented this algorithm, and made a pull request for the same. While testing my code, I came across a bug in the function `MinimalFaithfulPermutationDegree` , which is a function that relies on the subgroup lattice to calculate the Minimal Faithful Permutation Degree $\mu(G)$ for any finite group G.

The algorithm for *semi-simple* groups also relies on another algorithm that computes $\mu(G)$ for any *almost simple* group G, given the simple group $S \leq G \leq Aut(S)$. This is called *"lifting"* in the paper by *Das and Thakkar*. The lifting process is specific to the Isomorphism type of S. The main paper gives polynomial time algorithms for the lifting process in each case. The previously known methods, were not fast enough, and so, there was no algorithm for $\mu(G)$ for almost simple group G implemented in GAP till now. I wrote code for the lifting process for the required infinite families of simple groups S when S is a matrix group (namely $PSL_n(q)$ and $P\Omega_{2d}^{+}(q)$), and left the finite and non-matrix cases to the already existing `MinimalFaithulPermutationDegree` function with some pruning using Lagrange's theorem.

The lifting process for non-matrix groups requires knowledge of Lie Algebras, which is a topic I'm not very familiar with. Due to time constraints, the lack of appropriate functionalities in GAP, and some vital references being unavailable to me, I had to resort to directly using `MinimalFaithulPermutationDegree` for computing $\mu(A)$ albeit with a bit of pruning using Lagrange's theorem, where $S \leq A \leq Aut(S)$ and S is a non matrix simple group of lie type.

Finally, I completed the main function that computes the minimal permutation degree for any semi-simple group. I wrote some tests to compare my functions with existing ones, and the data that was available for permutation degree of simple groups. I then made a pull request for the same.

Outline of Algorithm:

1. Given input G , find the minimal normal subgroups.

2. For any minimal normal subgroup $N \trianglelefteq G$, find its direct decomposition $N = S_1 \times S_2 \times S_3 \dots S_l$.
   Each
   S_i will be isomorphic to the same simple group in its classification.
3. Compute for each N, $A = \{\phi_g | g \in N_G(S_1)\} \cong N_G(S)/C_G(S)$ where $\phi_g : S_1 \to S_1$ is given by $\phi_g(s) = g^{-1}sg$.
4. Compute $\mu(G, N) = \mu(A)$ using the lifting algorithm :
   a. If S_1 is not one of the exceptional cases, set $\mu(A) = \mu(S)$ and continue. Otherwise, proceed to second step.
   b. Express A as a group of automorphisms of $S \cong S_1$ where S is the "type" of simple group that S_1 is. For matrix groups, S is usually the projective group of some matrix group R .
   c. Construct a very specific generating set of R.
   d. *Lift* the automorphisms λ_i of S generating A , to automorphisms α_i of R which will generate a group isomorphic to A
   e. For each α , express it as $\alpha = \mathfrak{i} \cdot \mathfrak{d} \cdot \mathfrak{g} \cdot \mathfrak{f}$ and check if $\mathfrak{g}$ is trivial.
      If it is is trivial for each
      α , then $\mu(A) = \mu(S_1)$, otherwise, use the value specified in the paper for $\mu(A)$
5. Compute $\mu(G) = \sum_N l_N \, \mu(A_N)$.

Summary of Work done

1. Wrote a function that computes the minimal generating set of a group using chief series and appropriate tests.
2. Found the cause of a bug in SageMath related to the `libgap` interface to GAP.
3. Wrote a function to find the minimal faithful permutation degree for simple groups and appropriate tests.
4. Found a bug in `MinimalFaithfulPermutationDegree`
5. Wrote a function to find the minimal faithful permutation degree for almost simple groups using the lifting process described in the paper.
6. Wrote a function to find the minimal faithful permutation degree for semi-simple groups as described in the paper and tests for it.

Challenges

1. The mathematical nature of algorithms, despite making the project very interesting, was a barrier for efficient development for someone like me who isn't that proficient in group theory.
2. The simple groups have two different schemes of classification based on their isomorphism type. There is also a lot of variation for the meaning of different terms in different research papers.
3. Some of the less popular groups like $P\Sigma U_n(q)$, $\Gamma G_n(q)$ have almost no information on them online and the research papers which use these assume that the reader already knows their meaning.
4. Most of the contributors to GAP do it as a hobby.
   So for getting feedback and reviews on my pull requests after the initial commit, I had to wait for a long time.
5. A lot of the functions in GAP are undocumented, and have no references.
   For example, a randomised algorithm that finds the minimal generating set, namely
   `MinimalGeneratingSet` had no references. I had to read the code, which had almost no comments, to understand what the function was doing. Adding to that, the mathematical jargon that was unknown to me at that time, and it became a very difficult task.
6. The established team members at GAP are (rightly) critical of changes in the core library of GAP and prefer contributors writing a package instead.
7. The checks for a pull request made by an outsider can be initiated by team members at GAP only, and these, unlike in SageMath, aren't run automatically.

Things I wish I knew

1. GAP has all the minimal functionalities necessary for developers, like the `PageSource` function that shows the code for a function or class; the `?` prompt that shows the documentation of all documented functions with names starting with the text one puts after the `?` ; the line by line profiling facility in the package `profiling` , etc..

2. GAP doesn't have an edit distance and meta-data based search option. This makes it very difficult to discover new functions. For example, the `InducedAutomorphism` function returns an automorphism of $H \cong G$ that is equivalent to an automorphism $\phi(G)$, given the isomorphism $\psi : G \to H$ as the first argument. I was unable to find this function first when I needed it, and later found it mostly by chance. Reading the whole reference manual of GAP is the simpler, but tedious way around this.
3. The reviewers review the code thoroughly (for example, Max Horn) and (most of the times) make very helpful suggestions.
   Particularly, I received some really helpful suggestion from
   Christopher Jefferson on some initial pull requests.
4. The chances of a pull request getting reviewed are more at the start. Eventually, the interest in it is diminished.
5. The VS-code extension for GAP was also very helpful.

Future prospects

1. I plan on learning more about lie algebras and completing the lifting process when S is a non-matrix simple group of lie type.
2. I plan on creating a package containing these functions, as well as ones for automorphism decomposition ($\alpha = i \cdot d \cdot g \cdot f$) for simple groups of lie type.
3. I came to realise that GAP isn't perfect and lacks many functions, features, and documentation. Due to the laid back, but critical nature of GAP community, it takes a lot of time and effort for a beginner to contribute. I plan on continuing open-source contribution to GAP in the long term. I'll have to read a lot more discrete mathematics to do so. GAP, despite being mainly used for group theory, also has libraries for graph theory and automaton theory, among many other things. Thus, there is a lot of scope for long term open source contribution to GAP.