Clique and design finding in parallel using **GAP**, **GRAPE**, **DESIGN**, and **C**

Leonard Soicher

Queen Mary University of London

GAP Days, Aachen, October 2022

Introduction

I want to describe new hybrid **GAP**/**GRAPE**/**DESIGN**/**C** software to classify cliques in a given graph, which can also be used to classify block designs, and can make use of a high-performance computing (HPC) cluster.

GRAPE is a **GAP** package for computing with graphs together with associated groups of automorphisms. See https://gap-packages.github.io/grape/

DESIGN is a **GAP** package for constructing, classifying, and studying block designs. See https://gap-packages.github.io/design/

C is a fast, compiled programming language.

Computing cliques in **GRAPE**

All graphs in this talk are finite, undirected, with no loops and no multiple edges. Recall that a *clique* in a graph is a set of pairwise adjacent vertices.

Let Γ be a (simple) graph in **GRAPE**, with associated group $G \leq \mathrm{Aut}(\Gamma)$.

GRAPE functions can compute the maximal cliques of Γ or the cliques of given size in Γ, or the maximal cliques of given size in Γ, such that one such clique is determined or it is determined that no such cliques exist, or G-orbit generators of all such cliques are determined, or G-orbit representatives of all such cliques are determined.

Also, there is **GRAPE** functionality to determine a maximum clique in a graph, and hence to determine the clique number of that graph.

Cliques in graphs with vectors as vertex weights

GRAPE can also compute and classify the cliques with given vertex-weight sum in a vertex-weighted graph, where the weights are non-zero *d*-vectors of non-negative integers (satisfying certain conditions with respect to the group associated with the graph).

The **DESIGN** package for **GAP** can translate the construction and classification of many types of block designs into this type of weighted clique functionality, including block designs invariant under a specified group of automorphisms, as well as to construct and classify parallel classes and resolutions of block designs.

A **C** program

I have programmed in **C** a version of the **GRAPE** software to compute the cliques of given size in a graph (or more generally cliques of given vertex vector-weight sum), to handle the case when the associated group of automorphisms of the graph is trivial.

This can work in conjunction with **GAP** and **GRAPE**, as follows, to determine G-orbit generators for the k-cliques (cliques of size k) of a given graph Γ, where G is a given subgroup of the automorphism group of Γ. (We are now only considering the problem of determining/classifying k-cliques to avoid the technical details for a graph with vector vertex-weights.)

Outline of the method

First, a **GAP/GRAPE** function is used to perform a partial backtrack search, exploiting G, to generate a sequence

$$(P_1, A_1), (P_2, A_2), \ldots, (P_t, A_t),$$

where each P_i is a clique of size $\leq k$, and its corresponding A_i is a set of vertices of Γ disjoint from P_i, where if G_i is the (setwise) stabilizer of P_i in G, the following hold:

- for each i, A_i is G_i invariant and each vertex of A_i is adjacent to each vertex of P_i
- each G-orbit of k-cliques of Γ has at least one representative consisting of some P_i extended by $k - |P_i|$ elements belonging to the corresponding A_i
- for each i, we have $|P_i| = k$, or the image of G_i acting on A_i has order $\leq c_1$ (the default is $c_1 = 1$, i.e. the image is trivial) and A_i has size $\leq c_2$ (the use of c_2 is to attempt load balancing in the case of parallel computation).

Next, for each tuple (P_i, A_i), we use our **C** program (in parallel on the QMUL Apocrita cluster if desired) to determine the set B_i of all the cliques of size $k - |P_i|$ in the subgraph of Γ induced on A_i (so $\{P_i \cup B \mid B \in B_i\}$ is the set of cliques of size k of Γ which are extensions of P_i by elements from A_i).

Now let C be the union of the sets B_i. Then C is a set of k-cliques of Γ containing at least one representative from every G-orbit of k-cliques of Γ.

If required, one could then use Steve Linton's `SmallestImageSet` program or the **GAP** package **Images** (authored by Chris Jefferson, Markus Pfeiffer, Rebecca Waldecker, and Eliza Jonauskyte) or the **GAP** package **Vole** (authored by Mun See Chang, Chris Jefferson, and Wilf Wilson) to determine a subset of C containing exactly one representative from each G-orbit of the k-cliques of Γ.

Remarks

The **GAP**/**GRAPE** function to determine the sequence $(P_1, A_1), (P_2, A_2), \ldots, (P_t, A_t)$ is essentially the clever backtrack clique classifier in **GRAPE**, but with each (P_i, A_i) output (to a file) and its corresponding subtree not explored.

The (A_i, P_i) need not be pairwise G-inequivalent (with respect to the action `OnTuplesSets`), but if it is not too computationally expensive, one could use **Images** or **Vole** to remove any redundant G-orbit generators from $(P_1, A_1), (P_2, A_2), \ldots, (P_t, A_t)$.

One can also direct the **C** program to stop if a single clique of size k of Γ is found, if that is all that is required.

An example

Let Γ be the noncollinearity graph of the Cohen-Tits near octagon. Then Γ has 315 vertices, is regular with degree 304, and has automorphism group isomorphic to $J_2{:}2$, of order 1209600.

I was interested to determine the clique number of Γ. After some experimental work, I believed this clique number to be 90, so I set out to show that Γ has a clique of size 90, but none of size 91.

To determine the cliques of size $k := 90$, I used $G := \mathrm{Aut}(\Gamma)$ as the associated group of automorphisms of Γ, took $c_1 := 1$, $c_2 := 240$, and then my **GAP/GRAPE** function produced $t = 14814$ cases $(P_1, A_1), (P_2, A_2), \ldots, (P_t, A_t)$ as specified above, and which turned out to be pairwise G-inequivalent. The computation this far took about 5 minutes on my i5 Linux laptop.

These 14814 cases were then run in parallel using my **C** program on 200 cores of the QMUL Apocrita cluster, taking about 5 hours of user time. These computations returned a total of 31 90-cliques of Γ, but these all turned out to be G-equivalent.

An easy calculation in Γ then showed that a clique of size 90 is maximal, and so Γ has no clique of size 91.

Conclusion

More experimentation is required on the use of the constants c_1 and c_2 defined above.

More experimentation is required on the use of the **Images** and **Vole GAP** packages in this work.

Finally, if you have a problem requiring the determination of cliques in a graph, or block designs with given properties, please speak to me at this workshop. My programs and I may be able to help!