Graphical and Computational Representation of Groups

Alain Bretto and Luc Gillibert

Université de Caen, GREYC CNRS UMR-6072,Campus II, Bd Marechal Juin BP 5186, 14032 Caen cedex, France. {alain.bretto,lgillibe}@info.unicaen.fr

Abstract. An important part of the computer science is focused on the links that can be established between group theory and graph theory. CAYLEY graphs can establish such a link but meet some limitations. This paper introduces a new type of graph associated to a group: the G-graphs. We present an implementation of the algorithm constructing these new graphs. We establish a library of the most common G-graphs, using GAP and the `SmallGroups` library. We give some experimental results with GAP and we show that many classical graphs are G-graphs.

1 Introduction

The group theory, especially the finite group theory, is one of the main parts of modern mathematics. Groups are objects designed for the study of symmetries and symmetric structures, and therefore many sciences have to deal with them. Graphs can be interesting tools for the study of groups, a popular representation of groups by graphs being the CAYLEY graphs, an extended research has been achieved in this direction [1]. The regularity and the underlying algebraic structure of CAYLEY graphs make them good candidates for applications such as optimizations on parallel architectures, or for the study of interconnection networks [4]. But these properties are also a limitation: many interesting graphs are not CAYLEY graphs.

The purpose of this paper is to introduce a new type of graph – called G-graphs – constructed from a group and to present an algorithm to construct them. This algorithm is used for establishing some experimental results and for finding which graphs are G-graphs and which graphs are not. In fact, G-graphs, like CAYLEY graphs, have both nice and highly-regular properties. Consequently, these graphs can be used in any areas of science where CAYLEY graphs occur. Moreover many usual graphs, as the cube, the hypercube, the cuboctahedral graph, the Heawood's graph and lots of others, are G-graphs. We prove that some generic and infinite families of graphs, such as the complete bipartite graphs, are G-graphs. We establish a catalogue of the most common G-graphs, and for each of these graphs we exhibit the corresponding group, using the GAP's `SmallGroups` library. We also show that some non-vertex-transitive graphs, such as the Gray graph and the Ljubljana graph, are also G-graphs. In contrast, notice that CAYLEY graphs are always vertex-transitive.

The G-graphs are very informative about the groups from which thy are constructed: (1) they can be used for studying subgroups, via the correspondence

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 343–350, 2004.

between induced subgraphs and subgroups, and (2) the orders of the elements of a group can be read in the corresponding graph. In [2], it was shown that two isomorphic groups give two isomorphic graphs and that two abelian groups are isomorphic if and only if their associated graphs are themselves isomorphic. Thus, G-graphs can establish a link between the graph isomorphism problem and the abelian group isomorphism problem. But what happens for non abelian groups? We give some answers with an experimental simulation on all the groups of small order, again using GAP and the `SmallGroups` library.

2 Basic Definitions

We define a graph $\Gamma = (V; E; \epsilon)$ as follows::

- V is the set of vertices and E is the set of edges.
- ϵ is a map from E to $P_2(V)$, where $P_2(V)$ is the set of subsets of V having 1 or 2 elements.

In this paper graphs are finite, i.e., sets V and E have finite cardinalities. For each edge a, we denote $\epsilon(a) = [x; y]$ if $\epsilon(a) = \{x, y\}$ with $x \neq y$ or $\epsilon(a) = \{x\} = \{y\}$. If $x = y$, a is called *loop*. The set $a \in E, \epsilon(a) = [x; y]\}$ is called *multiedge* or *p-edge*, where p is the cardinality of the set. We define the degree of x by $d(x) = |\{a \in E, x \in \epsilon(a)\}|$.
In this paper, groups are also finite. We denote the unit element by e. Let G be a group, and let $S = \{s_1,\ s_2, \ldots,\ s_k\}$ be a nonempty subset of G. S is a *set of generators* of G if any element $\theta \in G$ can be written as a product $\theta = s_{i_1} s_{i_2} s_{i_3} \ldots s_{i_t}$ with $i_1, i_2, \ldots i_t \in \{1,\ 2, \ldots,\ k\}$. We say that G is *generated* by $S = \{s_1,\ s_2, \ldots,\ s_k\}$ and we write $G = \langle s_1, s_2, \ldots, s_k\rangle$. Let H be a subgroup of G, we denote Hx instead of $H\{x\}$. The set Hx is called *right coset* of H in G. A subset T_H of G is said to be a *right transversal* for H if $\{Hx, x \in T_H\}$ is precisely the set of all cosets of H in G.

3 Graph Group Process

Let (G, S) be a group with a set of generators $S = \{s_1,\ s_2,\ s_3 \ldots\ s_k\}$, $k \geq 1$. For any $s \in S$, we consider the left action of the subgroup $H = \langle s\rangle$ on G. Thus, we have a partition $G = \bigsqcup_{x \in T_s} \langle s\rangle x$, where T_s is a right transversal of $\langle s\rangle$. The cardinality of $\langle s\rangle$ is $o(s)$ where $o(s)$ is the order of the element s. Let us consider the cycles

$$(s)x = (x, sx, s^2x, \ldots, s^{o(s)-1}x)$$

of the permutation g_s: $x \longmapsto sx$. Notice that $\langle s\rangle x$ is the support of the cycle $(s)x$. Also ust one cycle of g_s contains the unit element e, namely $(s)e = (e, s, s^2, \ldots, s^{o(s)-1})$. We now define a new graph denoted $\Phi(G; S) = (V; E; \epsilon)$ as follows:

- The vertices of $\Phi(G;S)$ are the cycles of g_s, $s \in S$, i.e., $V = \sqcup_{s\in S} V_s$ with $V_s = \{(s)x, x \in T_s\}$.
- For all $(s)x, (t)y \in V$, $\{\langle s\rangle x, \langle t\rangle y\}$ is a p-edge if $card(\langle s\rangle x \cap \langle t\rangle y) = p$, $p \geq 1$.

Thus, $\Phi(G;S)$ is a k-partite graph and any vertex has a $o(s)$-loop. We denote $\tilde{\Phi}(G;S)$ the graph $\Phi(G;S)$ without loop. By construction, one edge stands for one element of G. One can remark that one element of G labels several edges. Both graphs $\Phi(G;S)$ and $\tilde{\Phi}(G;S)$ are called *graph from group* or *G-graph* and we say that the graph is *generated* by the groups $(G;S)$. Finally, if $S = G$, the G-graph is called a *canonic graph*.

3.1 Algorithmic Procedure

The following procedure constructs a graph from the list of the cycles of the group:

```
Group_to_graph_G(L)
for all s in L
    Add s to S
    for all s' in L
        for all x in s
            for all y in s'
                if x=y then Add (s,s') to A
```

An implementation of this procedure has been written in C++: we call it `Gro2gra`. The complexity of our implementation is $O(n^2 \times s^2)$ where n is the order of the group G and s is the cardinal of the family S. An other procedure constructs the vertices, that is the list of the cycles from the group G and the family S. The implementation of this procedure requires a high-level language that can manipulate groups: we use GAP Release 4.3 (Groups, Algorithms, and Programming) [5]. The following algorithm uses two functions:

1. `c_cycles`: computes a list of lists of lists, in fact the list of the lists of the cycles of each element s.
2. `fx`: writes the results of `c_cycles` in a file.

Only the procedure `c_cycles` is interesting:

```
InstallGlobalFunction (c_cycles, function(G, ga)
local ls1,ls2,gs,k,x,oa,a,res,G2;
res:=[]; G2:=List(G);
for a in ga do
      gs:=[]; oa:=Order(a)-1;
      ls2:=Set([]);
      for x in G do
            if not(x in ls2) then
                  ls1:=[];
                  for k in [0..oa] do;
                        Add(ls1, Position(G2, (a^k)*x));
```

```
                        AddSet(ls2, (a^k)*x);
                od; Add(gs, ls1);
            fi;
      od; Add(res, gs);
od;
return res; end);
```

For each s, $T_s = \{x_1, x_2, \ldots, x_j\}$, the right transversal of $\langle s \rangle$, is computed during the construction of the cycles $(s)x_i$. For this purpose, all the elements y of the cycle $\langle s \rangle x_i$ are added to the set `ls2`, then the procedure chooses an element x_{i+1} in G that does not appears in `ls2`, computes $(s)x_{i+1}$ and adds the elements of $\langle s \rangle x_{i+1}$ in `ls2`. The operation is repeated until all the elements of G are in `ls2`. Then, the set `ls2` is emptied; a new $s \in S$ is chosen and the operation is repeated. The second function, `fx`, is only here for the human's interface.

3.2 Example

In order to compute the cycles of the graph $\Phi(C_2 \times C_2; S)$ with $S = C_2 \times C_2$, we only have to call the function:
`fx(AbelianGroup([2,2]),AbelianGroup([2,2]),"c2c2");`
The procedure `fx` creates the following cycles in a file `c2c2`:
`(1)(2)(3)(4)(1 2)(3 4)(1 3)(2 4)(1 4)(2 3)`
Then, the program `Gro2gra` creates the following edges:

```
"(1)"--"(1 2)" "(1)"--"(1 3)" "(1)"--"(1 4)" "(2)"--"(1 2)"
"(2)"--"(2 4)" "(2)"--"(2 3)" "(3)"--"(3 4)" "(3)"--"(1 3)"
"(3)"--"(2 3)" "(4)"--"(3 4)" "(4)"--"(2 4)" "(4)"--"(1 4)"
"(1 2)"--"(1 3)" "(1 2)"--"(2 4)" "(1 2)"--"(1 4)"
"(1 2)"--"(2 3)" "(3 4)"--"(1 3)" "(3 4)"--"(2 4)"
"(3 4)"--"(1 4)" "(3 4)"--"(2 3)" "(1 3)"--"(1 4)"
"(1 3)"--"(2 3)" "(2 4)"--"(1 4)" "(2 4)"--"(2 3)"
```

These graph is shown in Fig. 1.

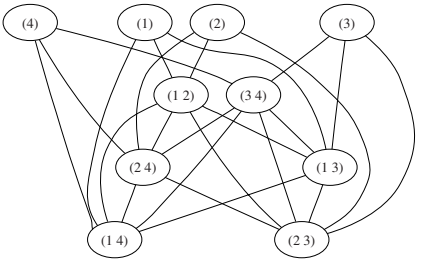

Fig. 1. $C_2 \times C_2$

4 Experimental Results

4.1 The Dihedral Group, the Generalized Quaternion Group, and the Product of Two Cyclic Groups

Let the dihedral group D_{2n} be the group of presentation:

$$\langle r, s \mid r^n = e, s^2 = e, sr = r^{n-1}s \rangle$$

Proposition 1: For $S = \{r, s\}$, the graph D_{2n}, $\tilde{\Phi}(D_{2n}; S)$ of the dihedral group is the complete bipartite graph $K_{2,n}$.
See Fig. 2 for an example.

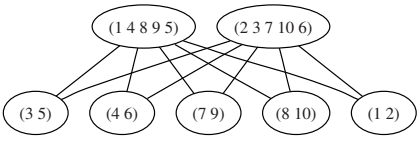

Fig. 2. $\tilde{\Phi}(D_{10}; \{a, b\})$

Let the generalized quaternion group Q_n be the group of presentation:

$$\langle a, b \mid a^{2n} = e, b^2 = a^n, ab = ba^{2n-1} \rangle$$

Proposition 2: For $S = \{a, b\}$, the graph Q_n, $\tilde{\Phi}(Q_n; S)$ of the generalized quaternion group is the complete double-edged bipartite graph $K^2_{2,n}$.
See Fig. 3 for an example.

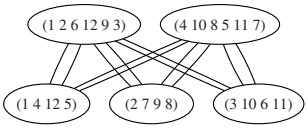

Fig. 3. $\tilde{\Phi}(Q_3; \{a, b\})$

Let $C_n \times C_k$ be the product of two cyclic groups. Such a product is generated by two elements, a and b, with $a^n = b^k = e$. More precisely, $C_n \times C_k$ is the group of presentation:

$$\langle a, b \mid a^n = e, b^k = e, ab = ba \rangle$$

Proposition 3: For $S = \{a, b\}$, the graph $C_n \times C_k$, $\tilde{\Phi}(C_n \times C_k; S)$ of the product of two cyclic groups, is the complete bipartite graph $K_{n,k}$.
See Fig. 4 for an example.

4.2 How to Recognize a *G*-Graph

Given a G-graph Γ, an interesting problem is how to find a group G and a family S such that $\tilde{\Phi}(G; S)$ isomorphic to Γ. If both G and S exist, we say that

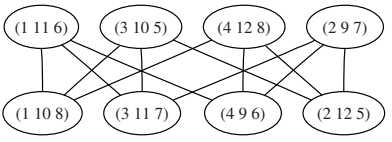

Fig. 4. $\tilde{\Phi}(C_3 \times C_3; \{a, b\})$

Γ is a G-graph. Here, we use the `SmallGroups` library from GAP. This library gives us access to all groups of certain small orders. The groups are sorted by their orders and they are listed up to isomorphism. Currently, the library contains the groups of order at most 2000 except 1024 (423 164 062 groups). In this section, we prove that many usual graphs are G-graph and we exhibit their corresponding groups.

The cube – Let us consider the skeleton of a cube. It is a graph with 8 vertices and 12 edges. All vertices are of degree 3 and the graph is bipartite. Suppose the cube is a G-graph $\tilde{\Phi}(G; S)$. Then the corresponding group G is of order 12 and is generated by a family S of cardinality 2, because the graph is bipartite. The alternate group with 12 elements, A_4, subgroup of S_4, is generated by the two cycles $(1, 2, 3)$ and $(1, 3, 4)$. Let S be the family $\{(1, 2, 3), (1, 3, 4)\}$. If we compute the graph $\tilde{\Phi}(A_4; S)$ with our algorithm, then we find the graph depicted in Fig 5.

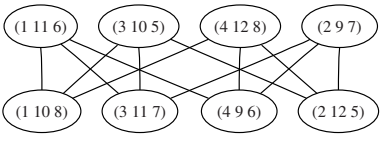

Fig. 5. $\tilde{\Phi}(A_4; S)$

It is easy to check that this graph is isomorphic to the cube. Thus, the cube is a G-graph as expected .

The hypercube – Let us consider the skeleton of an hypercube of dimension 4. It is a graph with 16 vertices and 32 edges. All vertices are of degree 4 and the graph is bipartite. Suppose the hypercube is a G-graph $\tilde{\Phi}(G; S)$. Then the corresponding group G is of order 32 generated by a family S of cardinal 2, because the graph is bipartite. The order of the elements of the family S must be 4 because the vertex degree is 4. If we look at the library `SmallGroups` we find 51 groups of order 32. Only 7 groups of order 32 can be generated by two elements of order 4: the groups number 2, 6, 10, 11, 13, 14 and 20. If we compute the corresponding graphs with our algorithm we find that `SmallGroup(32,6)` matchs (see Fig. 6).

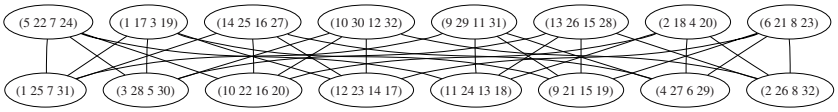

Fig. 6. $\tilde{\Phi}(\texttt{SmallGroup}(32,6); S)$

Others G-graphs – We give here some examples of G-graphs. The corresponding groups are indicated between parenthesis:

1. Bipartite complete graphs ($G = C_n \times C_k$, $S = \{(1,0)(0,1)\}$)
2. The 3-prism ($G = C_3 \times C_3$, $S = \{(1,0)(0,1)\}$)
3. The cuboctahedral graph ($G = C_2 \times C_2 \times C_2$, $S = \{(1,0,0),(0,1,0),(0,0,1)\}$)
4. The square (G is the Klein's group, $G = \{e, a, b, ab\}$, and $S = \{a, b\}$)
5. The cube ($G = A_4$, $S = \{(1,2,3),(1,3,4)\}$)
6. The hypercube ($G =$`SmallGroup(32,6)`, $S = \{f1, f1 * f2\}$)
7. The generalized Petersen's graph $P_{8,3}$
   ($G =$`SmallGroup(24,3)`, $S = \{f1, f1 * f2\}$)
8. The 2×2 grid on a torus ($G = Q_2$, $S = \{a, b\}$)
9. The 3×3 grid on a torus ($G = D_6$, $S = \{s \in G, Ordre(S) = 2\}$)
10. The 4×4 grid on a torus ($G =$`SmallGroup(32,6)`, $S = \{f1, f1 * f2\}$)
11. The Heawood's graph ($\langle a, b \mid a^7 = b^3 = e, ab = baa\rangle$, $S = \{b, ba\}$)
12. The Pappus's graph
    ($G = \langle a, b, c \mid a^3 = b^3 = c^3 = e, ab = ba, ac = ca, bc = cba\rangle$, $S = \{b, c\}$)
13. The Mobius-Kantor's graph ($G =$`SmallGroup(24,3)`, $S = \{f1, f1 * f2\}$)
14. The Gray graph ($G =$`SmallGroup(81,7)`, $S = \{f1, f2\}$)
15. The Ljubljana graph ($G =$`SmallGroup(168,43)`, $S = \{f1, f1 * f2 * f4\}$)

4.3 Couples of Non-isomorphic Groups Giving Isomorphic Graphs

One of the main goals of the G-graphs was originally the study of the graph isomorphism problem. A result in [2] says that two isomorphic groups give two isomorphic graphs. Another result in the same paper says that two abelian isomorphic groups are isomorphic if and only if their associated graphs are isomorphic. But what happens with non abelian groups? With our implementation of the algorithm `Gro2gra` and the `SmallGroups` library of GAP, it is possible to check automatically all the couples of non-isomorphic groups up to the order 100 in only a few days of computation.
Only couples of groups having the same number of elements of the same order can give isomorphic graphs. Such couples are be called "suspicious" couples in the table bellow. All isomorphisms are tested with Nauty [7]. Only orders with a least one suspicious couple are listed.

It is easy to see that only a few percent of suspicious couples give isomorphic graphs. The conclusion is that G-graphs are informative enough in the majority of the cases to allow the identification of the group by the graph. Finally we can notice that the couples of groups giving isomorphic graphs share the same properties:

They are non-simple, non-perfect, solvable and super-solvable.

Order	Number of groups	Number of suspicious couples	Non-isomorphic groups giving isomorphic graphs
16	14	7	1
27	5	2	2
32	51	66	3
48	52	20	3
54	15	6	2
64	267	1425	24
72	50	1	0
80	52	23	7
81	15	15	13
96	231	345	12
100	16	1	1

References

1. L. Babai. *Automorphism groups, isomorphism, reconstruction.* Chapter 27 of Handbook of combinatorics, 1994.
2. Alain Bretto and Alain Faisant, *A new graph from a group*, To appear in Compte rendu de l'academie des sciences, Paris, 2003.
3. John F. Humphreys, *"A course in Group Theory"*, Oxford University Press, 1997.
4. G. Cooperman and L. Finkelstein and N. Sarawagi. *Applications of Cayley Graphs.* Appl. Algebra and Error-Correcting Codes. Springer Verlag. Lecture Notes in Computer Sciences, Vol. 508 1991, 367–378.
5. The GAP Team, (06 May 2002), *"GAP - Reference Manual"*, Release 4.3, http://www.gap-system.org.
6. Joseph Lauri and Raffaele Scapellato, *Topics in Graphs Automorphisms and Reconstruction*, London Mathematical Society Student Texts, 2003.
7. Brendan D. McKay, Computer Science Department, Australian National University, (1981), *"Practical graph isomorphism"*, Congressus Numerantium 30, p. 45-87.