UNIVERSITAT DE BARCELONA

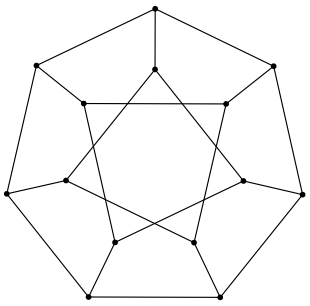
Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

MONOID GRAPHS AND GENERALIZED PETERSEN GRAPHS

Autor: Ernest Vidal i Garcia



Director: Dr. Kolja Knauer
Realitzat a: Departament de Matemàtica Aplicada

Barcelona, 13 de juny de 2023

Abstract

First, a wide definition of Cayley graphs is presented. We focus on the notion of *monoid graph*: a graph is a monoid graph if it is isomorphic to the underlying graph of the Cayley graph $\mathrm{Cay}(M, C)$ of some monoid M with some connection set $C \subseteq M$. Secondly, the family of Generalized Petersen Graphs $G(n, k)$ is presented. We study the open question whether every Generalized Petersen Graph is a monoid graph, and we focus on the smallest one for which the question remains unanswered: $G(7, 2)$. Finally, we explore the feasibility of using the computer to search for a possible monoid for $G(7, 2)$. We conclude that it is not viable to check all the possibilities with the proposed algorithms. Nevertheless, we are able to provide a computer-assisted proof that if $G(7, 2)$ is a monoid graph then the connection set C does not have any invertible element.

Key words: Cayley graphs, monoid graphs, Generalized Petersen Graphs, endomorphisms

Resum

En primer lloc, presentem una definició àmplia dels grafs de Cayley. Ens centrem en el concepte *graf de monoide*: un graf és un graf de monoide si és isomorf al graf subjacent del graf de Cayley $\mathrm{Cay}(M, C)$ d'algun monoide M amb algun conjunt de connexió $C \subseteq M$. En segon lloc, presentem la família dels Grafs de Petersen Generalitzats $G(n, k)$. Estudiem la pregunta sense resposta de si tot Graf de Petersen Generalitzat és un graf de monoide, i ens centrem en el més petit pel qual no se sap: $G(7, 2)$. Finalment, estudiem la viabilitat d'utilitzar la computació per buscar un possible monoide per $G(7, 2)$. Concloem que no és viable comprovar totes les possibilitats amb els algoritmes proposats. Tot i així, som capaços de donar una demostració assistida per ordinador que si $G(7, 2)$ és un graf de monoide, llavors el conjunt de connexió C no té cap element invertible.

Paraules clau: grafs de Cayley, grafs de monoides, Grafs de Petersen Generalitzats, endomorfismes

Acknowledgments

I would like to thank my tutor Kolja for helping me all along the process; my friend Anna for her valuable comments on the revision of the document; and my family, especially my mother Àgueda and my brother Eduard, for giving me the energy and confidence to overcome the difficulties.

2020 Mathematics Subject Classification. 05C25, 20M32, 60L70, 08A35

Contents

Introduction

Algebraic graph theory is a branch of mathematics in which algebraic methods are applied to study graphs. One of the main tools to do so are *Cayley graphs*, which are used for representing simple algebraic structures like groups or monoids, and whose definition was already suggested by Arthur Cayley in 1878 [3]. The vertices of a Cayley graph represent the elements of the group, and the edges represent the abstract structure of the binary operation between.

It is well known that every graph is isomorphic to some induced subgraph of a group graph [1]. Nevertheless, not every graph is a group graph. For example, the Petersen graph is not a group graph, but it is a monoid graph [6].

Examples of non-monoid graphs have also been found [8] but, as far as we know, not a single example of a non-semigroup graph has been found up to date.

Generalized Petersen Graphs $G(n,k)$ constitute an interesting family of graphs, which was introduced by Coxeter [4] and named by Watkins [16]. There is a characterization of the Generalized Petersen Graphs that are group graphs [11], but it remains an open question whether all Generalized Petersen Graphs are monoid graphs.

One of the main objectives of this work is to collect existing knowledge on Generalized Petersen Graphs that are monoid graphs, and to study in depth the graph $G(7,2)$, which is the smallest Generalized Petersen Graphs for which the question remains unresolved. Finally, another objective is to explore the feasibility of using computation to study if it is a monoid graph.

In Chapter 1, we provide the definition of Cayley graphs and the general results for monoid graphs. In Chapter 2, the Generalized Petersen Graphs are presented and their relation with group graphs and monoid graphs is studied. In Chapter 3, we explain in detail the code that we developed in order to check if $G(7,2)$ is a monoid graph, as well as the obtained results. Finally, the conclusions of the work are presented in Chapter 4.

Chapter 1

Monoid graphs

In the first section of this chapter the notation is established and a wide definition of Cayley graphs is presented. In the second section we highlight the relation between monoid graphs and the endomorphism monoid of a graph, and present some well known results and characterizations of Cayley graphs of groups and of monoids. Finally, in the third section we collect some properties of monoid graphs that will be useful in later chapters.

1.1 Cayley graphs

A *graph* G consists of a vertex set V and an edge set E, $G = (V, E)$, where an edge is an unordered pair of vertices of G. A directed graph or *digraph* D consists of a vertex set V and an arc set A, $D = (V, A)$, where an arc, or directed edge, is an ordered pair of vertices of D. Usually, the words 'edge' and 'arc' are reserved for pairs of distinct vertices, whereas the word loop is used for edges or arcs from one vertex to itself. The terms *multigraph* and *multidigraph* are used to indicate that repeated edges or arcs are also considered. Repeated edges or arcs are called *parallel* edges or arcs; two arcs that join the same pair of vertices but on the opposite order are called *anti-parallel* arcs.

If $\{x, y\}$ is an edge, then we say that x and y are adjacent, or that y is a *neighbour* of x, $y \in N(x)$, and x is a neighbour of y, $x \in N(y)$. If (x, y) is an arc we say that x is an *in-neighbour* of y, $x \in N^-(y)$, and y is an *out-neighbour* of x, $y \in N^+(x)$. The *degree* $\delta(v)$ (resp. *in-degree* $\delta^-(v)$, *out-degree* $\delta^+(v)$) of a vertex v is the number of edges (resp. incoming arcs, outgoing arcs) of the vertex, which in general does not coincide with its number of neighbours (resp. in-neighbours, out-neighbours). A graph is *k-regular* (resp. *k-inregular*, *k-outregular*) if every vertex v has degree $\delta(v) = k$ (resp. in-degree $\delta^-(v) = k$, out-degree $\delta^+(v) = k$).

A set X with a binary operation $X \times X \to X$, $(a, b) \mapsto ab$, is called a *magma*.

The binary operation of a finite set $X = \{x_1, ..., x_n\}$ is usually given through the multiplication table, or Cayley table, which is a square matrix of size n such that in the position $[i][j]$ it has $x_i x_j$. A magma is called a *semigroup* S if the operation is associative. A semigroup is called a *monoid* M if it has a neutral element e. A monoid is called a *group* $\mathcal{G}$ if every element has an inverse. The notation $\subseteq$ is used to denote subsets while $\leq$ is used for subgroups, submonoids, subsemigroups and submagmas.

Definition 1.1. *Let X be a finite magma and $C \subseteq X$ a subset. The digraph* $\mathrm{Cay}(X, C) := (X, A)$ *with arc set A such that*

$$\forall x, y \in X: \qquad (x, y) \in A \iff \exists c \in C : xc = y$$

is called the Cayley digraph (or simply the Cayley graph) of X with connection set C . It can have loops and anti-parallel arcs but not parallel arcs.

The Cayley colored digraph, $\mathrm{Cay}_{\mathrm{col}}(X, C)$*, is obtained by assigning one color to every element c of the connection set C and then coloring each arc $(x, y) = (x, xc)$ with its corresponding color c. In this case multiple parallel arcs of different color are allowed.*

The underlying graph of the Cayley digraph, $\underline{\mathrm{Cay}}(X, C)$*, is the simple graph obtained by not considering colors, neglecting orientations (i.e. replacing every arc (x, y) with edge $\{x, y\}$), suppressing repeated edges and suppressing loops.*

The Cayley colored digraph is the graphic representation of X that contains most of its algebraic information. On the other hand, the underlying graph of the Cayley graph carries the least information but it is more natural from the perspective of graph theory, since it is a simple graph.

Example 1.2. The Cayley colored digraph of the monoid M defined by the following multiplication table with connection set $C = \{a, b, c\}$ is [Figure 1.1]:

Remark 1.3. The resulting digraph depends strongly on the chosen connection set, as seen in the following picture [Figure 1.2].

Definition 1.4. *Let G be a simple graph (uncolored, undirected and without loops). We say that G is a semigroup graph (resp. a monoid graph or group graph) if it is the underlying graph of the Cayley graph of a semigroup (resp. monoid or group). This is:*

$$G \text{ semigroup graph} \iff \exists \text{ semigroup } S, \exists \text{ subset } C \subseteq S\text{: } G \cong \underline{\mathrm{Cay}}(S, C)$$

$$G \text{ monoid graph} \iff \exists \text{ monoid } M, \exists \text{ subset } C \subseteq M\text{: } G \cong \underline{\mathrm{Cay}}(M, C)$$

$$G \text{ group graph} \iff \exists \text{ group } \mathcal{G}, \exists \text{ subset } C \subseteq \mathcal{G}\text{: } G \cong \underline{\mathrm{Cay}}(\mathcal{G}, C)$$

M	e	a	b	c	d	f
e	e	a	b	c	d	f
a	a	a	a	a	a	a
b	b	c	d	e	f	a
c	c	c	c	c	c	c
d	d	f	e	a	b	c
f	f	f	f	f	f	f

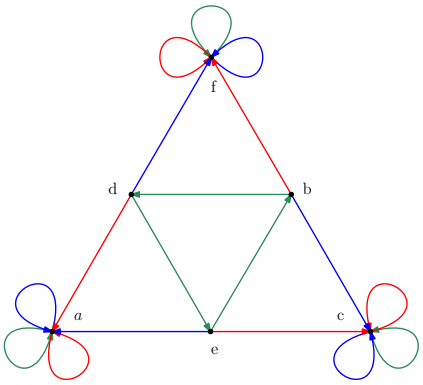


Figure 1.1: Left: multiplication table of the monoid M. Right: Cayley colored digraph $\mathrm{Cay}_{\mathrm{col}}(M, C)$ of the monoid M with connection set $C = \{a, b, c\}$.

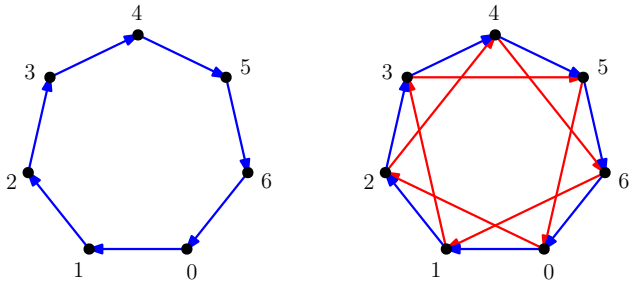


Figure 1.2: Left: $\mathrm{Cay}_{\mathrm{col}}(\mathbb{Z}_7, \{1\})$. Right: $\mathrm{Cay}_{\mathrm{col}}(\mathbb{Z}_7, \{1, 2\})$

Example 1.5. The graph $C_3 = K_3 = \Big(\{0,1,2\},\ \{(0,1),(0,2),(1,2)\}\Big)$ is a group graph, as it is, for example, the underlying graph of the Cayley graph of the group $\mathbb{Z}_3$ with the sum operation and connection set $C = \{1\}$. In this case there are other connection set that work [Figure 1.3].

There are non-monoid graphs. The smallest one we know of is the following graph, which was obtained by [8, Proposition 4.7.]:

Example 1.6. The disjoint union graph $K_4 \cup C_5$ is non-monoid [Figure 1.4].

Lemma 1.7. *Let X_1 and X_2 be magmas (resp. semigroups, monoids, groups) and $C_1 \subseteq X_1$ and $C_2 \subseteq X_2$ be subsets. If $C_1 \subseteq C_2$ and $X_1 \leq X_2$, then $G_1 = \mathrm{Cay}(X_1, C_1)$ is a subdigraph of $G_2 = \mathrm{Cay}(X_2, C_2)$, where $\leq$ stands for submagma (resp. subsemigroup, submonoid, subgroup).*

Proof. We know that $X_1 \subseteq X_2$, therefore $V_1 \subseteq V_2$, since the vertices of a Cayley graphs correspond to the elements of its magma. On the other hand, the fact

$\mathbb{Z}_3$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

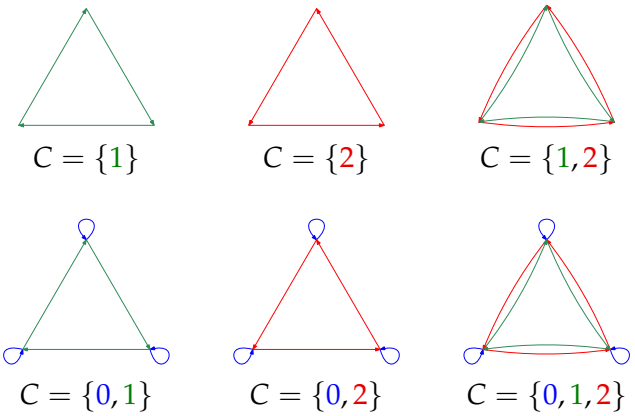


Figure 1.3: Left: sum table of the group $\mathbb{Z}_3$. Right: six possible ways to see the graph C_3 as a group graph of the group $\mathbb{Z}_3$ with different connection sets.

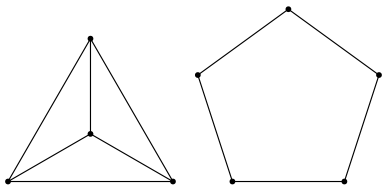


Figure 1.4: Graph $K_4 \cup C_5$

that the binary operation of X_1 and X_2 is the same, together with the hypothesis $C_1 \subseteq C_2$, implies that $A_1 \subseteq A_2$. □

1.2 The endomorphism monoid

A homomorphism is a structure-preserving map between two algebraic structures of the same type, and an endomorphism is a homomorphism from one algebraic structure to itself. For magmas, semigroups, monoids and groups, homomorphisms preserve the binary operation and its additional properties in each case.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. A mapping $\varphi : V_1 \to V_2$ is a *homomorphism* of graphs if it sends adjacent vertices of G_1 to adjacent vertices of G_2. If G_1 and G_2 are directed and/or colored graphs, we require homomorphisms to map any arc to an arc of the same direction and/or color. An invertible homomorphism φ whose inverse is also a homomorphism is an *isomorphism*.

Proposition 1.8. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs. A mapping $\varphi : V_1 \to V_2$ is an isomorphism if and only if it is a bijective homomorphism such that*

$\{u, v\} \in E_1 \iff \{\varphi(u), \varphi(v)\} \in E_2$.

Proof. Necessity. If φ, φ^{-1} are homomorphisms, we have that $\{u, v\} \in E_1 \Rightarrow \{\varphi(u), \varphi(v)\} \in E_2$ and $\{u', v'\} \in E_2 \Rightarrow \{\varphi^{-1}(u'), \varphi^{-1}(v')\} \in E_1$, which implies the condition $\{u, v\} \in E_1 \iff \{\varphi(u), \varphi(v)\} \in E_2$. Finally, the existence of inverse implies the bijectivity. *Sufficiency.* The right implication ($\Rightarrow$) of the condition directly implies that φ is an homomorphism. The bijectivity implies the existence of the inverse function φ^{-1}. Then, by applying φ^{-1} to both sides of the left implication ($\Leftarrow$) of the condition, we see that φ^{-1} is an homomorphism. □

Example 1.9. Consider graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of Figure 1.5. The homomorphism $\varphi : V_1 \to V_2$ that sends $1 \to a$, $2 \to b$ and $3 \to c$ is a bijective homomorphism of graphs that is not an isomorphism of graphs. This happens when E_2 has more edges than E_1.

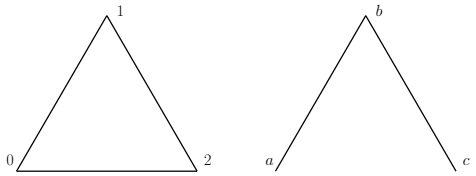


Figure 1.5: Graphs of Example 1.9. Left: G_1. Right: G_2

A homomorphism from a graph G to itself is an *endomorphism*, and an isomorphism from a graph G to itself is an *automorphism*.

Proposition 1.10. *Let $G = (V, E)$ be a graph. A mapping $\varphi : V \to V$ is an automorphism if and only if it is a bijective endomorphism.*

Proof. Necessity. An automorphism of G is necessarily a bijective endomorphism by applying Proposition 1.8. *Sufficiency.* Let φ be a bijective endomorphism. Since φ is bijective, it has finite order k ($\varphi^k = id$) and therefore there exists $\varphi^{-1} = \varphi^{k-1}$. Since φ is an endomorphism, $\{u, v\} \in E \Rightarrow \{\varphi(u), \varphi(v)\} \in E \Rightarrow \{\varphi^j(u), \varphi^j(v)\} \in E \ \forall j$. Then, $\{\varphi^{-1}(u), \varphi^{-1}(v)\} = \{\varphi^{k-1}(u), \varphi^{k-1}(v)\} \in E$, as we wanted to see. □

Given a graph G, the set of all its endomorphisms (with the composition of homomorphisms) form the endomorphism monoid of the graph, $\mathrm{End}(G)$, and the set of all its automorphisms form the automorphism group of the graph, $\mathrm{Aut}(G)$. Trivially, the endomorphism monoid of a graph contains its automorphism group, since every automorphism is an endomorphism: $\mathrm{Aut}(G) \leq \mathrm{End}(G)\ \forall G$.

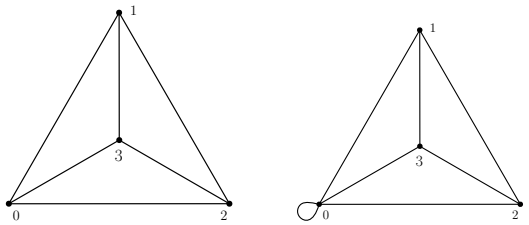
Let's see some basic properties of the endomorphism monoid of Cayley graphs.

Observation 1.11. *For a magma X and a subset $C \subseteq X$ we have:*

$$\mathrm{End}(\mathrm{Cay}_{col}(X, C)) \leq \mathrm{End}(\mathrm{Cay}(X, C))$$

Observation 1.12. *Contrary to what happens with orientations and colors, adding or removing loops has a critical impact on the endomorphism monoid: there exist X and C for which* $\mathrm{End}(\mathrm{Cay}(X, C)) \not\leq \mathrm{End}(\underline{\mathrm{Cay}}(X, C))$.

Example 1.13. Consider the following graphs G_1 and G_2 of Figure 1.6. Then con-



G_1 G_2

Figure 1.6: The graph K_3 without loops and with one added loop.

sider the mappings $\varphi_1 = [1, 2, 3, 0]$ and $\varphi_2 = [0, 0, 2, 3]$, which can also be written as follows:

$$\begin{array}{lll} \varphi_1: & 0 \mapsto 1 & \varphi_2: \quad 0 \mapsto 0 \\ & 1 \mapsto 2 & \quad\quad 1 \mapsto 0 \\ & 2 \mapsto 3 & \quad\quad 2 \mapsto 2 \\ & 3 \mapsto 0 & \quad\quad 3 \mapsto 3 \end{array}$$

Observe that $\varphi_1 \in \mathrm{End}(G_1)$ but $\varphi_1 \notin \mathrm{End}(G_2)$ and that $\varphi_2 \in \mathrm{End}(G_2)$ but $\varphi_2 \notin \mathrm{End}(G_1)$.

A key tool for working with Cayley graphs is the fact that the multiplication law of a monoid M is related to the endomorphism monoid $\mathrm{End}(D)$ of the Cayley digraphs $D = \mathrm{Cay}_{\mathrm{col}}(M, C)$ it defines, regardless of the chosen connection set C. Let's see this relation in depth.

Definition 1.14. *Let X be a semigroup and $a \in X$. Left-multiplication with a is the magma endomorphism $\lambda_a : X \to X$, $x \mapsto ax$.*

These are endomorphisms of the algebraic structure, and correspond to the rows of the multiplication table. When the structure has associativity, they also happen to be endomorphisms of the corresponding colored digraphs of the structure:

Lemma 1.15. *Let S be a semigroup, $C \subseteq S$ and $s \in S$. Left-multiplication λ_s is a color-preserving endomorphism of* $\mathrm{Cay}_{\mathrm{col}}(S, C)$.

Proof. Let (t, tc) be an arc of color c. Then, using associativity of S we have that its image by λ_s is the arc $(st, s(tc)) = (st, (st)c)$, which is also of color c. □

This means that when we have a semigroup digraph D, we can associate each vertex to its corresponding row in the multiplication table [8, Lemma 2.1.]:

Lemma 1.16. *Let S be a semigroup and $C \subseteq S$. Mapping every $s \in S$ to left-multiplication λ_s with s is a homomorphism from S to* $\mathrm{End}(\mathrm{Cay}_{\mathrm{col}}(S, C))$.

Proof. As we have seen, left-multiplication λ_s with s is an element of $\mathrm{End}(\mathrm{Cay}_{\mathrm{col}}(S, C))$. Clearly, $\lambda_{st} = \lambda_s \circ \lambda_t$, so this is a semigroup homomorphism. □

Observation 1.17. *The homomorphism of Lemma 1.16 is not necessarily injective.*

Example 1.18. Let's consider the right-zero semigroup, defined as $R_n = \{r_1, ..., r_n\}$ such that $r_i r_j = r_j \ \forall i, j$. We can see that $\mathrm{End}(\mathrm{Cay}_{\mathrm{col}}(R_4, R_4)) = id$ [Figure 1.7]. So in this semigroup mapping each element to left multiplication is not injective.

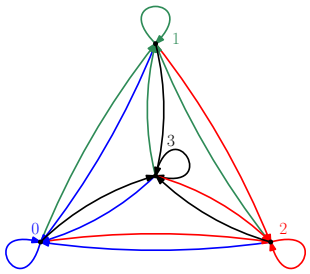


Figure 1.7: The Cayley graph $\mathrm{Cay}_{\mathrm{col}}(R_4, R_4)$.

For monoids, under the additional hypothesis that C is a generating set of M, we have a well-known stronger result [9, Theorem 7.3.7.]:

Lemma 1.19. *Let M be a monoid and be $C \subseteq M$ such that $M = \langle C \rangle$. Then left-multiplication yields an isomorphism from M to* $\mathrm{End}(\mathrm{Cay}_{\mathrm{col}}(M, C))$. *In other words,*

$$M \text{ monoid},\ C \subseteq M,\ \langle C \rangle = M \quad \Rightarrow \quad M \cong \mathrm{End}(\mathrm{Cay}_{\mathrm{col}}(M, C))$$

These results lead to a characterization of monoid digraphs. These characterizations are called Sabidussi-type characterizations, due to the first result on group graphs by Sabidussi [12, Lemma 4]:

Lemma 1.20. *A (colored, directed) graph $G = (V, A)$ is a (colored, directed) group graph if and only if* $\mathrm{Aut}(G)$ *has a subgroup $\mathcal{G}$ that acts regularly on G, i.e., for any two vertices $x, y \in V$ there exists a unique automorphism $\varphi \in \mathcal{G}$ such that $\varphi(x) = y$. In this case, G is a Cayley graph of the group $\mathcal{G}$.*

Corollary 1.21. *All group graphs are regular.*

Proof. By the Sabidussi Lemma 1.20, there are automorphisms between any pair of vertices of the graph. Since automorphisms preserve the degree of the vertices, the graph is regular. □

Furthermore, the connectivity of group graphs is ensured when the connection set is a generating set of the groups [7, Lemma 3.7.4.]:

Proposition 1.22. *A group graph $G \cong \underline{\mathrm{Cay}}(\mathcal{G}, C)$ is connected if and only if $\langle C \rangle = \mathcal{G}$.*

In the case of monoid digraphs, there is the following Sabidussi-type characterization [8, Lemma 2.4.]:

Lemma 1.23. *A directed graph $D = (V, A)$ is a monoid digraph if and only if there exists a vertex $e \in V$ and a submonoid $M \leq \mathrm{End}(D)$ such that for each $x \in V$ there is a unique $\varphi_x \in M$ with $\varphi_x(e) = x$. Moreover, φ_x satisfies that for each $(x, y) \in A$ there is a $(e, c) \in A$ such that $\varphi_x(c) = y$.*

Proof. Suppose that there is a vertex e and a submonoid $M \leq \mathrm{End}(G)$ satisfying this property. Let $C = \{\varphi \in M | (e, \varphi(e)) \in A\}$. We claim that there is an isomorphism $G \cong \mathrm{Cay}(M, C)$ given by $x \mapsto \varphi_x$. It is clear that this map is injective. It is also surjective: the preimage of $\varphi \in M$ is $\varphi(e)$. Now, if $(x, y) \in A$, there is an out-neighbour c of e with $\varphi_x \circ \varphi_c(e) = \varphi_x(c) = y$; thus $\varphi_x \circ \varphi_c = \varphi_y$. Since $\varphi_c \in C$, (φ_x, φ_y) is an arc of $\mathrm{Cay}(M, C)$. Conversely, if (φ_x, φ_y) is an arc of $\mathrm{Cay}(M, C)$ then $\varphi_x \circ \varphi = \varphi_y$ for some $\varphi \in C$. This implies that the arc $(e, \varphi(e)) \in A$ is mapped by φ_x to $(x, \varphi_x \circ \varphi(e)) = (x, y)$, which must be also in A, because φ_x is an endomorphism.

Conversely, suppose that $G = \mathrm{Cay}(M, C)$ for a monoid M and $C \subseteq M$. For each vertex x consider the endomorphism φ_x of G defined by left-multiplication by x. All these endomorphisms together form a submonoid $M' \leq \mathrm{End}(\mathrm{Cay}_{\mathrm{col}}(M, C)) \leq \mathrm{End}(G)$, by Lemma 1.16. Let e be the neutral element of M. It is clear that for any vertex x, the mapping φ_x is the only of them that maps e to x. Moreover, for any out-neighbour y of x there is an out-neighbour $c \in C$ of e with $xc = y$, so $\varphi_x(c) = y$. □

Finally, let's study the image of digraph endomorphisms. If we have an endomorphism f of some digraph $D = (V, A)$, we know some information about the image of f, denoted by $Im(f) = f(V)$, which will be useful later on:

Lemma 1.24. *Let $D = (V, A)$ be a finite colored digraph with n vertices and $f \in \mathrm{End}(D)$ an endomorphism of the colored digraph. Then, f is an automorphism if and only if its image is of size n:*

$$f \in \mathrm{End}(D) \qquad \Longrightarrow \qquad \left(\; f \in \mathrm{Aut}(D) \iff f(V) = V \; \right)$$

Proof. We are using that a bijective endomorphism is an automorphism (Proposition 1.10), and that on any two finite sets X and Y, $f : X \to Y$ is bijective if and only if X and Y have the same number of elements. □

Lemma 1.25. *Let $D = \mathrm{Cay}(M, C)$ be a monoid digraph and $\lambda_m \in \mathrm{End}(D)$ left-multiplication. Then, the image of the λ_m has no outgoing arcs, i.e., $\delta^+(Im(\lambda_m)) = 0$*

Proof. Let $x \in Im(\lambda_m)$. Then $\exists z$ such that $\lambda_m(z) = x$. Let (x, y) be an arc of color c. Then, using the associativity of the multiplication of M, we can see that y must also be in $Im(\lambda_m)$: $y = xc = \lambda_m(z)c = (mz)c = m(zc) = \lambda_m(zc) \in Im(\lambda_m)$. □

1.3 Properties of monoid graphs

In this section we collect the aspects of monoid graphs that show up on the actual drawing of the graph. First we look at the properties that a simple graph must have in order to be a monoid graph. Then we do the same for digraphs and colored digraphs.

1.3.1 Simple monoid graphs

The neutral element can be assumed not to be in the connection set:

Proposition 1.26. *If $G \cong \underline{\mathrm{Cay}}(M, C)$, it can be assumed without loss of generality that $e \notin C$.*

Proof. Having the neutral element in the connection set only adds one loop to all the vertices, which does not affect the underlying graph $\underline{\mathrm{Cay}}(M, C)$. □

It can also be assumed that C does not have the inverses of its elements (in case they exist and are different from themselves):

Proposition 1.27. *If $G \cong \underline{\mathrm{Cay}}(M, C)$ and $c \in C$ is invertible of order > 2, it can be assumed without loss of generality that $c^{-1} \notin C$.*

Proof. Having c and c^{-1} in the connection set only adds antiparallel arcs of color c^{-1} to all the arcs of color c. Since the direction and multiplicity of arcs is not important for $\underline{\mathrm{Cay}}(M, C)$, we can avoid this situation whenever we can, i.e. when $c^{-1} \neq c$. □

The neighbours of the neutral element can be used as connection set [8, Lemma 4.1.]:

Lemma 1.28. *Let M be a monoid, $C \subseteq M$ and the graph $G \cong \underline{\mathrm{Cay}}(M, C)$. Then $G \cong \underline{\mathrm{Cay}}(M, N(e))$.*

Note that this does not mean that all the neighbours of e are necessary. The minimal connection set might be a subset of $N(e)$:

Remark 1.29. If $G \cong \underline{\mathrm{Cay}}(M, C)$, then $C \subseteq N(e)$.

This sets a maximum order of C. In general, the minimum order of C is determined by the number of edges m in proportion to the vertices n of the graph, since there cannot be more edges than n times the order of C:

Proposition 1.30. *If $G \cong \underline{\mathrm{Cay}}(M, C)$, then:*

i. $|C| \leq \delta(e)$

ii. $|C| \geq m/n$

where n is the number of vertices and m is the number of edges.

Proof. Assuming that the neutral element e is not in C, we have:

i. This follows from 1.29. We are implicitly using that e does not have loops because $e \notin C$, so e is not an out-neighbour of himself.

ii. Any vertex of the Cayley colored multidigraph has by definition one outgoing arc of each color, so for each color c there are n edges of that color c. Since different colored arcs can be in the same edge, every $c \in C$ contributes at the most n edges in the underlying simple graph, so $m \leq n|C|$.

□

1.3.2 Monoid digraphs

First of all, the multidigraph is $|C|$-outregular, since every vertex has an outgoing arc of each color. The existence of neutral element plays a key role, since it sets remarkable restrictions to the possible orientations. In general, there can be parallel arcs, since the monoid can have $c \neq c'$ such that $sc = sc'$; and there can be loops, since it can have $c \neq e$ such that $sc = s$. But the neutral element has a distinctive behaviour:

Proposition 1.31. *If $D \cong \mathrm{Cay}(M, C)$, the neutral element has no loops.*

Proof. If e has a loop of color c, it must be $ec = e \Rightarrow c = e \in C$. If we assume that $e \notin C$, then neutral element has no loops. □

Proposition 1.32. *If $D \cong \mathrm{Cay}(M, C)$, the neutral element does not have parallel outgoing arcs.*

Proof. Let $c_1, c_2 \in C$ such that $ec_1 = ec_2$, then $c_1 = c_2$. □

The out-neighbours of e are now exactly the connection set of the Cayley graph:

Lemma 1.33. *If $D \cong \mathrm{Cay}(M, C)$, then $D \cong \underline{\mathrm{Cay}}(M, N^+(e))$.*

Proof. By definition, there is one outgoing arc for each element of the connection set C. Since there are no parallel outgoing arcs of e, each element of the connection set corresponds to one different out-neighbour of e. □

Under the addition hypothesis that C is a generating set there are stronger consequences:

Lemma 1.34. *If $D \cong \mathrm{Cay}(M, C)$ and $\langle C \rangle = M$, every vertex $v \neq e$ is reachable from e through a sequence of arcs.*

Proof. Using the correspondence between the digraph and the monoid, the fact that every vertex is reachable through a sequence of n arcs $a_1, \ldots, a_n$ (of colors $c_1, \ldots, c_n$) that start at e is equivalent to the fact that every element of M is equal to $ec_1 \ldots c_n$, which is the condition $\langle C \rangle = M$. □

1.3.3 Monoid colored digraphs

For colored digraphs there is actually a characterization of those that are monoid digraphs, by just looking at the colored arcs [2, Theorem 4.4].

Definition 1.35. *A colored path is a sequence of consecutive colored arcs. The label word L of a colored path is the sequence of colors of the arcs.*

Definition 1.36. *A propagating vertex p is a vertex such that, if there are two colored paths from p to x labeled by L_A and L_B then for any vertex v there are also two colored paths from v to x labeled by the same L_A and L_B.*

Theorem 1.37. *A graph is a monoid graph if and only if all the following conditions are fulfilled:*

1. *From every vertex of the graph there is exactly one outgoing arc of each color.*
2. *The graph has a vertex e such that every other vertex is reachable from e, e does not have parallel outgoing arcs and e is a propagating vertex.*

Finally, apart from the global properties of the directed colored graph, each component of each color also has additional properties to be fulfilled. The subdigraph constructed by keeping all the vertices and keeping only the arcs of one color is a 1-outregular digraph. The class of 1-outregular semigroup digraphs was studied by Zelinka [17].

Let D be a 1-outregular digraph. Each connected component $\mathcal{C}$ has exactly one cycle (possibly a loop), denoted by Z, with length $z(\mathcal{C})$. Then, we denote by $\ell(v)$ the length of the (unique) shortest directed path from vertex $v \in \mathcal{C}$ to the cycle Z, and by $\ell(\mathcal{C})$ the maximum length to the cycle among all the vertices of the connected component.

The following is a Zelinka-type characterization of 1-outregular monoid digraphs [8, Theorem 3.2]:

Theorem 1.38. *A 1-outregular digraph D is a monoid digraph if and only if D has a component $\mathcal{C}$ such that $z(\mathcal{D})$ devides $z(\mathcal{C})$ and $\ell(\mathcal{D}) \leq \ell(\mathcal{C})$ for all components $\mathcal{D}$ of D. Moreover, the component $\mathcal{C}$ is the component of the neutral element e and the element e is one that attains $\ell(\mathcal{C})$.*

Chapter 2

Generalized Petersen Graphs G(n,k)

In the first section of this chapter, we present the Generalized Petersen Graphs, which is a particularly interesting family of graphs, and we set the notation. In the second section we see some results that might help to understand which Generalized Petersen Graphs are monoid graphs. In the third section we focus especially on $G(7,2)$, which is the smallest of them for which the question remains open.

2.1 Generalized Petersen Graphs

Generalized Petersen graphs are defined as follows [16]:

Definition 2.1. *Let k, n be integers such that $0 < k < \frac{n}{2}$. The* Generalized Petersen graph $G(n,k)$ *is the cubic graph with vertex set*

$$V = \{u_0, u_1, \ldots, u_{n-1}, v_0, v_1, \ldots, v_{n-1}\},$$

and with edges set E consisting of all those of the form

$$[u_i, u_{i+1}] \quad [u_i, v_i] \quad [v_i, v_{i+k}]$$

where i is an integer and all subscripts are to be read modulo n.

Observation 2.2. *All Generalized Petersen Graphs are cubic, i.e.,* 3*-regular.*

Example 2.3. The Petersen graph itself is the graph $G(5,2)$ [Figure 2.1]

Example 2.4. Among the generalized Petersen graphs are the n-prism $G(n,1)$, the Dürer graph $G(6,2)$, the Möbius-Kantor graph $G(8,3)$, the dodecahedron $G(10,2)$, the Desargues graph $G(10,3)$ and the Nauru graph $G(12,5)$ [Figure 2.2].

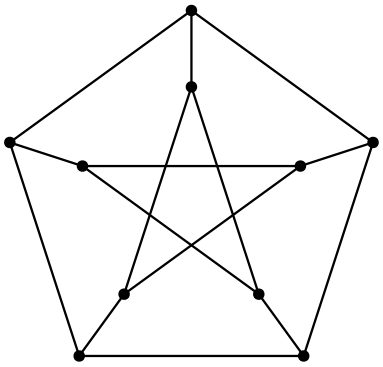


Figure 2.1: The Petersen graph $G(5,2)$.

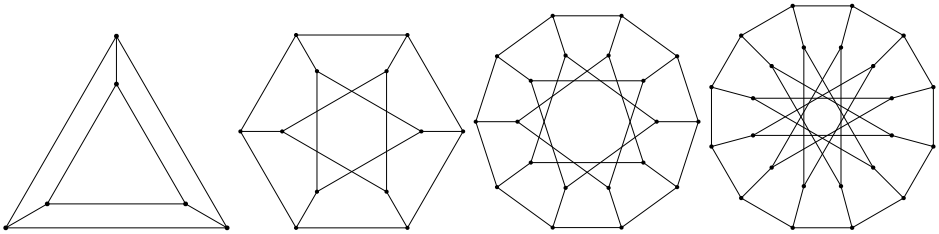


Figure 2.2: From left to right: the triangular prism $G(3,1)$, the Dürer graph $G(6,2)$, the Desargues graph $G(10,3)$ and the Nauru graph $G(12,5)$.

The *dihedral group* $D_{2\cdot n}$ is the group of symmetries of the regular n-gon. When drawing the Generalized Petersen Graphs as suggested in Figures 2.1 and 2.2 it becomes evident that these symmetries are also symmetries of the graphs.

Observation 2.5. *If G is a Generalized Petersen Graph, then the dihedral $D_{2\cdot n}$ is a subgroup of $Aut(G)$: $D_{2\cdot n} \leq Aut(G)$.*

The automorphism group of all Generalized Petersen Graphs depending on n and k was determined in 1971 by Frucht, Graver, and Watkins [5]. Here we just present the case $k^2 \neq \pm 1 \pmod n$ as an example, since it is the case for $G(7,2)$.

Proposition 2.6. *If G is a generalized Petersen graph of the form $G(n,k)$ where (n,k) is not equal to $(4,1)$, $(5,2)$, $(8,3)$, $(10,2)$, $(10,3)$, $(12,5)$, or $(24,5)$, and $k^2 \neq \pm 1 \pmod n$, then $Aut(G) \cong D_{2\cdot n}$.*

The range of indices n and k in the definition is chosen to avoid some isomorphic graphs, but there are still some other isomorphisms among the Generalized Petersen Graphs [14].

Theorem 2.7. *$G(n,k) \cong G(n,l)$ if and only if either*

(i) $l \equiv \pm k (mod\ n)$, *or*

(ii) $gcd(n,k) = gcd(n,l) = 1$ *and* $kl \equiv \pm 1 (mod\ n)$.

The first case is implicitly considered by our definition of $G(n,k)$ since we only pick $k < \frac{n}{2}$. In the following table (Table 2.1) we represent all Generalized Petersen Graphs included by our definition above the dotted line. For each n up to $n = 16$ we count how many isomorphic classes $\cong^i$ of $G(n,k)$ there are. The pairs of isomorphic graphs are represented in different colors.

n	1	2	3	4	5	6	7	k
16	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^4$	$\cong^3$	$\cong^5$	$\cong^6$	
15	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^4$	$\cong^5$	$\cong^6$	$\cong^2$	
14	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^4$	$\cong^3$	$\cong^5$		
13	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^3$	$\cong^4$	$\cong^2$		
12	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^4$	$\cong^5$			
11	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^3$	$\cong^2$			
10	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^4$				
9	$\cong^1$	$\cong^2$	$\cong^3$	$\cong^2$				
8	$\cong^1$	$\cong^2$	$\cong^3$					
7	$\cong^1$	$\cong^2$	$\cong^2$					
6	$\cong^1$	$\cong^2$						
5	$\cong^1$	$\cong^2$						
4	$\cong^1$							
3	$\cong^1$							

Table 2.1: Isomorphism classes $\cong^i$ among Generalized Petersen Graphs $G(n,k)$ up to $n = 16$.

2.2 Monoids and Generalized Petersen Graphs

There is a characterization of the Generalized Petersen graphs that are group graphs [11]:

Theorem 2.8. *$G(n,k)$ is a group graph if and only if $k^2 \cong 1\ (mod\ n)$.*

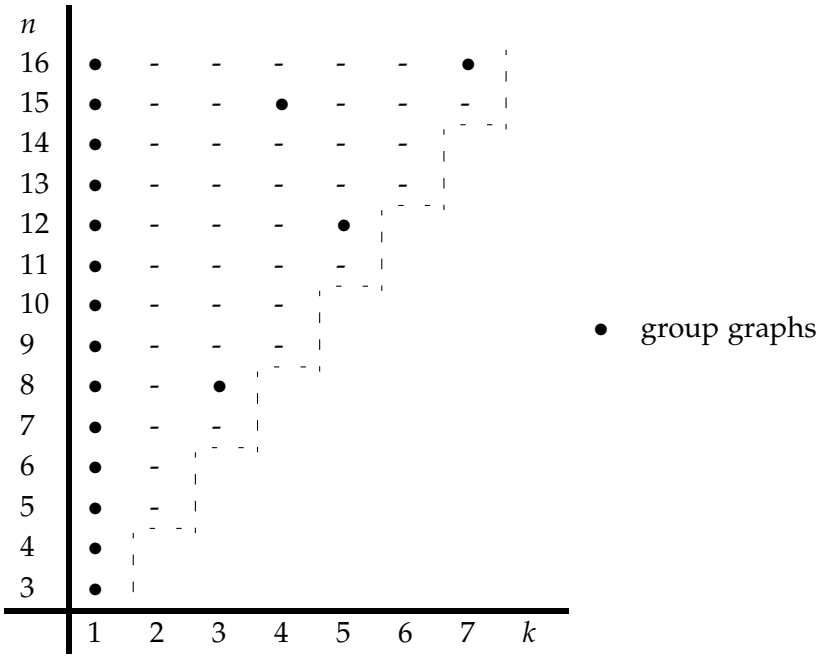


Table 2.2: Generalized Petersen Graphs that are group graphs (•).

The following table (Table 2.2) includes all the Generalized Petersen graphs up to $n = 16$ and indicates if they are a group graph (•). If $G(n,k_1) \cong G(n,k_2)$, we only represent the graph with smallest k and leave a blank space for the other one.

As we can see in the Table 2.2, the Petersen graph is the first Generalized Petersen Graph that is not a group graph.

Nevertheless it is a monoid graph [6, Proposition 3.6.]:

Example 2.9. $G(5,2)$ has different possible semigroups and monoids, as shown in Table 2.3. S and M are unions of $\mathbb{Z}_6$ and the *null semigroup* $N_{[6,9]}$, for which $ab = 9$ for all $a,b \in \{6,7,8,9\}$. S' and M' are unions of the dihedral group $D_{2\cdot 3}$ and the *null semigroup* $N_{[6,9]}$. Then, $G(5,2) \cong \underline{\text{Cay}}(S,\{1,6\}) \cong \underline{\text{Cay}}(M,\{1,6\}) \cong \underline{\text{Cay}}(S',\{0,4,8\}) \cong \underline{\text{Cay}}(M',\{0,4,8\})$ [Figure 2.3].

The next Generalized Petersen Graph, $G(6,2)$, is also a monoid graph.

Example 2.10. $G(6,2)$ is a monoid graph, and a possible monoid is shown in Table 2.4. M is the union $M = A \cup A'$ with $A = \mathbb{Z}_6$ and $A' = \mathbb{Z}_3 \times L_2$, where $L_k := L_{\mathbb{Z}_k}$ is the left-zero semigroup with k elements, for which $l_i l_j = l_i$ for all $l_i, l_j \in L_k$. This construction is based on [6, Theorem 3.11]. Then, $G(6,2) \cong \underline{\text{Cay}}(M,\{1,6\}) \cong \underline{\text{Cay}}(M,\{1,8\}) \cong \underline{\text{Cay}}(M,\{1,10\})$ [Figure 2.4].

S	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	0	7	8	6	9
2	2	3	4	5	0	1	8	6	7	9
3	3	4	5	0	1	2	6	7	8	9
4	4	5	0	1	2	3	7	8	6	9
5	5	0	1	2	3	4	8	6	7	9
6	9	9	9	9	9	9	9	9	9	9
7	9	9	9	9	9	9	9	9	9	9
8	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9

M	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	0	7	8	6	9
2	2	3	4	5	0	1	8	6	7	9
3	3	4	5	0	1	2	6	7	8	9
4	4	5	0	1	2	3	7	8	6	9
5	5	0	1	2	3	4	8	6	7	9
6	6	6	6	6	6	6	9	9	9	9
7	7	7	7	7	7	7	9	9	9	9
8	8	8	8	8	8	8	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9

S'	0	1	2	3	4	5	6	7	8	9
0	5	4	3	2	1	0	8	7	6	9
1	2	3	4	5	0	1	8	6	7	9
2	1	0	5	4	3	2	7	6	8	9
3	4	5	0	1	2	3	7	8	6	9
4	3	2	1	0	5	4	6	8	7	9
5	0	1	2	3	4	5	6	7	8	9
6	9	9	9	9	9	9	9	9	9	9
7	9	9	9	9	9	9	9	9	9	9
8	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9

M'	0	1	2	3	4	5	6	7	8	9
0	5	4	3	2	1	0	8	7	6	9
1	2	3	4	5	0	1	8	6	7	9
2	1	0	5	4	3	2	7	6	8	9
3	4	5	0	1	2	3	7	8	6	9
4	3	2	1	0	5	4	6	8	7	9
5	0	1	2	3	4	5	6	7	8	9
6	6	6	6	6	6	6	9	9	9	9
7	7	7	7	7	7	7	9	9	9	9
8	8	8	8	8	8	8	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9

Table 2.3: Multiplication tables of different semigroups and monoids for $G(5,2)$. From left to right, from top to bottom: S, M, S', M'.

For the moment, there is not a characterization for the Generalized Petersen graphs that are monoid graphs. Nevertheless we have some useful information about the hypothetical connection set.

Proposition 2.11. *If a cubic graph is a monoid graph, then* $|C| = 2$ *or* $|C| = 3$.

Proof. We know that in general $\frac{m}{n} \leq |C| \leq \delta(e)$ (Proposition 1.30). By the handshake lemma, a cubic graph has $m = \frac{3}{2}n$ edges and it obviously has $\delta(e) = 3$. This implies that there are only two possibilities for the order of the generating set C: $|C| = 2$ or $|C| = 3$. □

Corollary 2.12. *If* $G(n,k) \cong \underline{\mathrm{Cay}}(M,C)$, *then* $|C| = 2$ *or* $|C| = 3$.

Proof. By construction, any Generalized Petersen Graph is cubic (Observation 2.2). □

Under the additional hypothesis that $\langle C \rangle = M$, stronger results can be found. For the specific case of Generalized Petersen Graphs with $\langle C \rangle = M$ and with $|C| = 2$ there is the following remarkable characterization [6, Theorem 3.13.]:

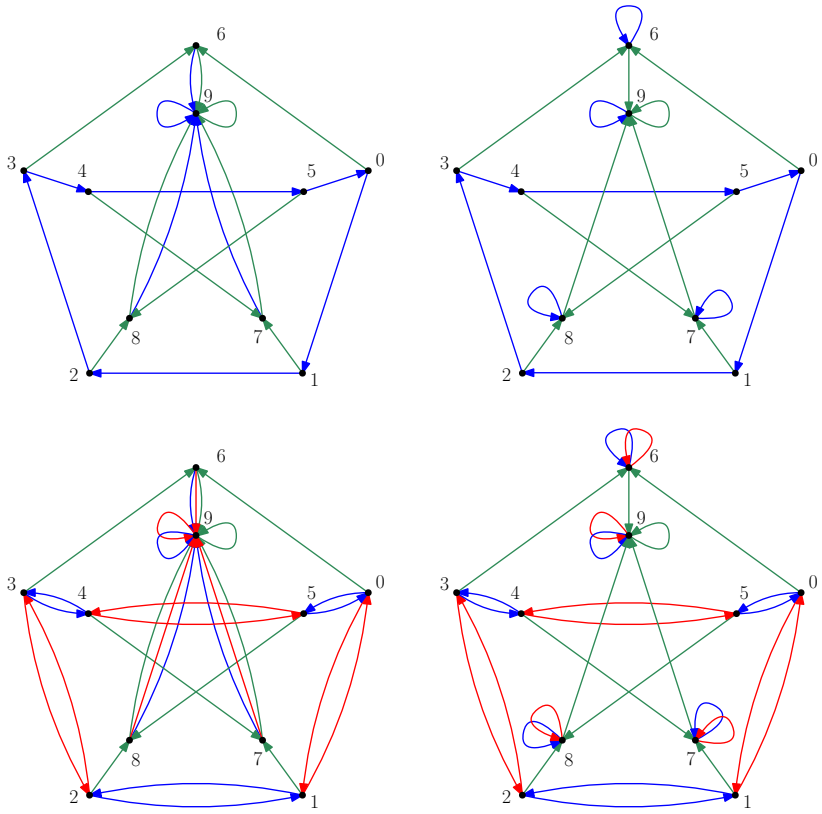


Figure 2.3: From left to right, from top to bottom: the Cayley colored digraphs $\mathrm{Cay}_{\mathrm{col}}(S, \{1,6\})$, $\mathrm{Cay}_{\mathrm{col}}(M, \{1,6\})$, $\mathrm{Cay}_{\mathrm{col}}(S', \{0,4,8\})$ and $\mathrm{Cay}_{\mathrm{col}}(M', \{0,4,8\})$.

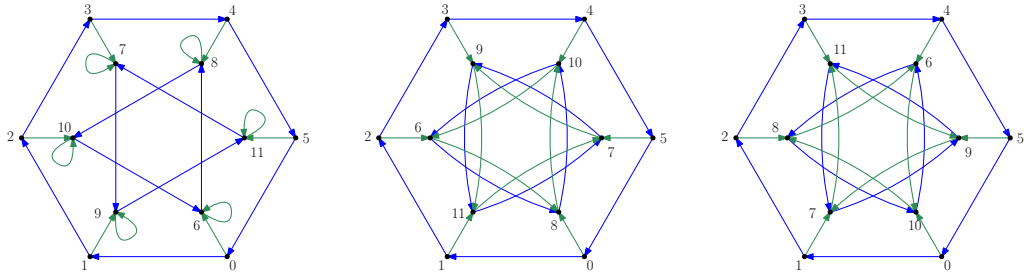


Figure 2.4: The colored digraphs $\mathrm{Cay}_{\mathrm{col}}(M, \{1,6\})$, $\mathrm{Cay}_{\mathrm{col}}(M, \{1,8\})$ and $\mathrm{Cay}_{\mathrm{col}}(M, \{1,10\})$.

Theorem 2.13. *The Generalized Petersen Graph $G(n,k)$ is a monoid graph $\mathrm{Cay}(M,C)$ with $\langle C\rangle = M$ and $|C| = 2$ if and only if one of the following holds:*

(a) $(n,k) = (5,2)$ *(Petersen graph),*

(b) $k^2 \equiv 1 \pmod{n}$, *or*

M'	0	1	2	3	4	5	$(0,l_0)$	$(0,l_1)$	$(1,l_0)$	$(1,l_1)$	$(2,l_0)$	$(2,l_1)$
0	0	1	2	3	4	5	$(0,l_0)$	$(0,l_1)$	$(1,l_0)$	$(1,l_1)$	$(2,l_0)$	$(2,l_1)$
1	1	2	3	4	5	0	$(1,l_1)$	$(1,l_0)$	$(2,l_1)$	$(2,l_0)$	$(0,l_1)$	$(0,l_0)$
2	2	3	4	5	0	1	$(2,l_0)$	$(2,l_1)$	$(0,l_0)$	$(0,l_1)$	$(1,l_0)$	$(1,l_1)$
3	3	4	5	0	1	2	$(0,l_1)$	$(0,l_0)$	$(1,l_1)$	$(1,l_0)$	$(2,l_1)$	$(2,l_0)$
4	4	5	0	1	2	3	$(1,l_0)$	$(1,l_1)$	$(2,l_0)$	$(2,l_1)$	$(0,l_0)$	$(0,l_1)$
5	5	0	1	2	3	4	$(2,l_1)$	$(2,l_0)$	$(0,l_1)$	$(0,l_0)$	$(1,l_1)$	$(1,l_0)$
$(0,l_0)$	$(0,l_0)$	$(1,l_0)$	$(2,l_0)$	$(0,l_0)$	$(1,l_0)$	$(2,l_0)$	$(0,l_0)$	$(0,l_0)$	$(1,l_0)$	$(1,l_0)$	$(2,l_0)$	$(2,l_0)$
$(0,l_1)$	$(0,l_1)$	$(1,l_1)$	$(2,l_1)$	$(0,l_1)$	$(1,l_1)$	$(2,l_1)$	$(0,l_1)$	$(0,l_1)$	$(1,l_1)$	$(1,l_1)$	$(2,l_1)$	$(2,l_1)$
$(1,l_0)$	$(1,l_0)$	$(2,l_0)$	$(0,l_0)$	$(1,l_0)$	$(2,l_0)$	$(0,l_0)$	$(1,l_0)$	$(1,l_0)$	$(2,l_0)$	$(2,l_0)$	$(0,l_0)$	$(0,l_0)$
$(1,l_1)$	$(1,l_1)$	$(2,l_1)$	$(0,l_1)$	$(1,l_1)$	$(2,l_1)$	$(0,l_1)$	$(1,l_1)$	$(1,l_1)$	$(2,l_1)$	$(2,l_1)$	$(0,l_1)$	$(0,l_1)$
$(2,l_0)$	$(2,l_0)$	$(0,l_0)$	$(1,l_0)$	$(2,l_0)$	$(0,l_0)$	$(1,l_0)$	$(2,l_0)$	$(2,l_0)$	$(0,l_0)$	$(0,l_0)$	$(1,l_0)$	$(1,l_0)$
$(3,l_1)$	$(2,l_1)$	$(0,l_1)$	$(1,l_1)$	$(2,l_1)$	$(0,l_1)$	$(1,l_1)$	$(2,l_1)$	$(2,l_1)$	$(0,l_1)$	$(0,l_1)$	$(1,l_1)$	$(1,l_1)$

$\cong$

$\cong$

M	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	3	4	5	6	7	8	9	10	11
1	1	2	3	4	5	0	9	8	11	10	7	6
2	2	3	4	5	0	1	10	11	6	7	8	9
3	3	4	5	0	1	2	7	6	9	8	11	10
4	4	5	0	1	2	3	8	9	10	11	6	7
5	5	0	1	2	3	4	11	10	7	6	9	8
6	6	8	10	6	8	10	6	6	8	8	10	10
7	7	9	11	7	9	11	7	7	9	9	11	11
8	8	10	6	8	10	6	8	8	10	10	6	6
9	9	11	7	9	11	7	9	9	11	11	7	7
10	10	6	8	10	6	8	10	10	6	6	8	8
11	11	7	9	11	7	9	11	11	7	7	9	9

Table 2.4: Table of a monoid M for $G(6,2)$. Top: illustrative construction of the table. Bottom: isomorphic table changing the names of the elements.

(c) $k^2 \equiv \pm k \pmod{n}$

The condition $\langle C \rangle = M$ can be dropped under certain circumstances [6, Proposition 3.19, Proposition 3.20, Conjecture 3.18.]:

Theorem 2.14. *Let* $1 \leq k \leq n/2$ *such that* $\gcd(n,k) = 1$. *The Generalized Petersen Graph* $G(n,k)$ *is a monoid graph* $\mathrm{Cay}(M,C)$ *with* $|C| = 2$ *if and only if one of the following holds:*

(a) $(n,k) = (5,2)$ *(Petersen graph),*

(b) $(n,k) = (10,3)$ *(Desargues graph), or*

(c) $k^2 \equiv 1 \pmod{n}$

Theorem 2.15. *Let* $1 \leq k \leq n/2$ *such that* $\gcd(n,k) = 1$ *or such that* $n/\gcd(n,k)$ *is odd. The Generalized Petersen Graph* $G(n,k)$ *is a monoid graph* $\mathrm{Cay}(M,C)$ *with* $|C| = 2$ *if and only if one of the following holds:*

(a) $(n,k) = (5,2)$ *(Petersen graph),*

(b) $k^2 \equiv 1 \pmod{n}$*, or*

(c) $k^2 \equiv \pm k \pmod{n}$

Conjecture 2.16. *The Generalized Petersen Graph* $G(n,k)$ *is a monoid graph* $\mathrm{Cay}(M,C)$ *with* $|C| = 2$ *if and only if one of the following holds:*

(a) $(n,k) = (5,2)$ *(Petersen graph),*

(b) $(n,k) = (10,3)$ *(Desargues graph),*

(c) $k^2 \equiv 1 \pmod{n}$*, or*

(d) $k^2 \equiv \pm k \pmod{n}$

To sum up, the current knowledge on Generalized Petersen graphs that are monoid graphs is represented in Table 2.5.

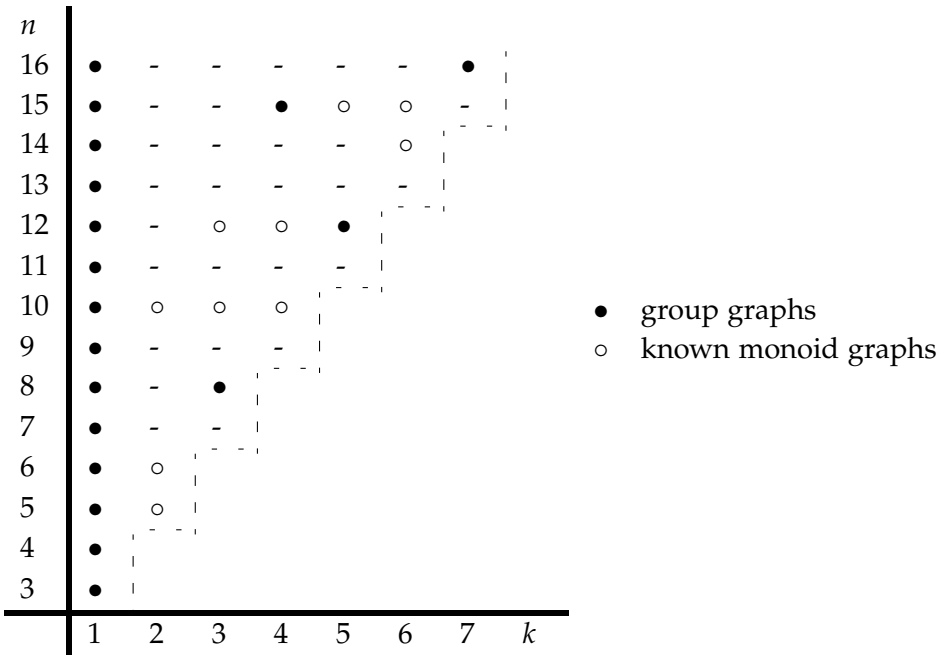


Table 2.5: Generalized Petersen Graphs that are group graphs (•) and known Generalized Petersen Graphs that are monoid graphs (○).

2.3 G(7,2)

Let us focus on the graph $G(7,2)$ which, as we have seen, is the smallest Generalized Petersen Graph that it is not known if it is a monoid graph or not.

First of all, we know that the connection set has exactly 3 elements:

Proposition 2.17. *If* $G(7,2) \cong \underline{\text{Cay}}(M,C)$, *then* $|C| = 3$.

Proof. Proposition 2.12 says $|C|$ is 2 or 3. Then, by Theorem 2.14 (or also Theorem 2.15) we have that the case $|C| = 2$ is not possible for $n = 7$ and $k = 2$. So if $G(7,2)$ is a monoid graph it must have $|C| = 3$. □

Corollary 2.18. *If* $G(7,2) \cong \underline{\text{Cay}}(M,C)$, *then* $|C| = N(e)$.

Proof. We know that $C \subseteq N(e)$ (Remark 1.29) and $\delta(e) = 3$ (Observation 2.2) so Proposition 2.17 implies that $|C| = N(e)$. □

We also know some restrictions on the invertible elements of C:

Lemma 2.19. *If* $G(7,2) \cong \underline{\text{Cay}}(M,C)$, *then* C *does not contain any invertible element of order* > 2.

Proof. Since $G(7,2) \cong \underline{\text{Cay}}(M,C)$, then $C = N(e)$ because $|C| = 3$ (Proposition 2.17), $C \subseteq N(e)$ (Remark 1.29) and $|N(e)| = 3$ (Observation 2.2). Then, if $c \in C = N(e)$ is invertible of order > 2, $c^{-1} \neq c$ and $c^{-1} \in N(e)$. Then $c^{-1} \in C$ but this is a contradiction because C is assumed not to have the inverses of its invertible elements of order > 2 (Proposition 1.27). □

Lemma 2.20. *If* $G(7,2) \cong \underline{\text{Cay}}(M,C)$, *then* C *cannot have 2 invertible elements.*

Proof. Let $c, d \in C$ be invertible. We have that $c \neq d^{-1}$ and $d \neq c^{-1}$ (Proposition 1.27). Then $X = \langle c,d \rangle$ is a group, therefore $H = Cay(\langle c,d \rangle, \{c,d\})$ is a Cayley graph of a group contained in $G(7,2)$ (Lemma 1.7). H is a group graph, so it is regular (Corollary 1.21). Since $G(7,2)$ is not a group graph (Theorem 2.8), H is a proper subgraph of $G(7,2)$; then H must be 2-regular. Since the connection set is a generating set of the group, H is connected (Proposition 1.22). So H is connected and 2-regular, which implies that it must be a cycle. Then, there are two possibilities for the group X, since $X \leq \text{Aut}(H)$ (Lemma 1.20):

- X is a cyclic group. But then its Cayley graph is a cycle only if it is generated by exactly one element (!)
- X is a dihedric group. We also know that the group X is a submonoid of M and $M \leq \mathrm{End}(G(7,2))$ (Lemma 1.23), so X is a subgroup of $\mathrm{Aut}(G(7,2)) = D_{2\cdot 7}$ (Proposition 2.6). But it is well known that $D_{2\cdot m} < D_{2\cdot n} \iff m|n$, so $D_{2\cdot 7}$ cannot have a dihedric subgroup X since 7 is prime (!)

□

An invertible element with order 2 is called *involution*. So we have seen that either C has exactly one invertible element and it is an involution or C does not have any invertible elements.

Chapter 3

Computational search: is G(7,2) a monoid graph?

In this chapter we explore the feasibility of using the computer to check if $G(7,2)$ is a monoid graph. In the first section, we explore the possibilities that the theory gives us to search all the possible monoids. We design two possible approaches to the problem, explained in detail in the second and third sections. In the fourth section we explore the case of non-generated monoid graphs.

3.1 The plan

We have seen that the graph $G(7,2)$ is the smallest Generalized Petersen Graph for which it is unknown whether it is a monoid graph or not. We want to use computation to search for a monoid M and a subset $C \subseteq M$ such that $G(7,2) \cong \underline{\mathrm{Cay}}(M,C)$.

If it is a monoid graph, the connection set C must fulfill the following properties:

i. $|C| = 3$ so $C = N(e)$ (Proposition 2.17, Corollary 2.3)

ii. C does not have invertible elements of order > 2 (Lemma 2.19)

iii. C has at most one involution (Lemma 2.20)

We also know that, if it is a monoid graph, there exists a digraph D of which $G(7,2)$ is its underlying graph such that M is a submonoid of $\mathrm{End}(D)$:

i. There exists a 3-outregular multiorientation D (with loops) of the graph such that $M \cong \{f_i\}_{i=0,\dots,13} \leq \mathrm{End}(D)$ where $f_i(e) = v_i \ \forall v_i \in V$ (Lemma 1.23). Moreover, if $\langle C \rangle = M$, then $M \cong \langle f_1, f_2, f_3 \rangle$, where $f_j(e) = c_j \in C = N(e)$ (Corollary 2.3).

ii. There exists a 3-outregular colored multiorientation D_{col} (with loops) of the graph such that $M \cong \{f_i\}_{i=0,\dots,13} \leq \mathrm{End}(D_{col})$ where $f_i(e) = v_i \ \forall v_i \in V$ (Lemma 1.23). Moreover, if $\langle C \rangle = M$, then $M \cong \mathrm{End}(D_{col})$ (Lemma 1.19).

The search space is so vast that we focus on the case $\langle C \rangle = M$, so from now on we will assume that the connection set C is a generating set of M unless stated otherwise. In conclusion, all these results and considerations induce two different approaches for the computational search of M:

Approach 1: For each possible multiorientation of the graph without loops, for each the candidates of neutral element e, for each way of adding loops, compute $\mathrm{End}(D)$ and for each subset of 3 elements $f_1, f_2, f_3 \in \mathrm{End}(D)$ such that $N(e) = \{c_1, c_2, c_3\} = \{f_1(e), f_2(e), f_3(e)\}$ check if $G(7,2) \cong \underline{\mathrm{Cay}}(\langle c_1, c_2, c_3 \rangle, \{c_1, c_2, c_3\})$.

Approach 2: For each possible multiorientation of the graph without loops, for each the candidates of neutral element e, for each way of adding loops, for each way of coloring the arcs, compute $\mathrm{End}(D_{col})$ and check if $G(7,2) \cong \underline{\mathrm{Cay}}(M, C)$ with $M = \{f_i(e) | f_i \in \mathrm{End}(D_{col})\}$ and $C = N(e)$.

Since these endomorphisms f_i correspond to left multiplication with v_i, they can be seen as the rows of the multiplication table of M. When considering $M = \{f_i\}_{i=0,\dots,13}$ as a monoid, their order does not matter, but when considering them as the rows of the multiplication table, they have to be put in the right order so that f_i is exactly the row i, since it sends e to v_i.

Note that the characterization of colored digraphs that are monoid digraphs of Theorem 1.37 is not viable to check on the computer, since determining whether the vertex e is a propagating vertex could in principle need an infinite number of steps, since there are an infinite number of possible paths between any pair of vertices.

Finally, another theoretical approach would be to compute all monoids M of $N = 14$ elements and check all the subsets C of 3 elements, and then check if $G(7,2) \cong \underline{\mathrm{Cay}}(M, C)$. But this is also not viable since the number of monoids grows fast with N [13, `https://oeis.org/A058129`]. In fact, the total number of nonisomorphic monoids of $N = 14$ elements is unknown. A lower bound to the number of monoids of N elements is the number of semigroups of $M = N - 1$ elements, since you can always add a neutral element to obtain a monoid. In this case, this lower bound is also unknown since the number of nonisomorphic semigroups also grows fast with M [13, `https://oeis.org/A027851`].

The code is written in python. Although this programming language is slower than other options, it allows us to use SageMath [15], which is a free open-source mathematics software system that includes useful libraries to work with digraphs, including the object class `DiGraph()` (`class sage.graphs.digraph.DiGraph`). This will allow us to focus on the mathematical aspects of the code and the optimization of the algorithms themselves.

The code can be found in `https://github.com/ernestv98/TFG_Mat.git`. Along the following sections its structure is explained and some parts are shown in a bit more detail.

3.2 Approach 1: digraph endomorphisms

3.2.1 Case A: C has no invertible elements

First of all, we want to create a file that contains a list of all the possible multi-orientations of the graph $G(7,2)$, stored in a compact way.

The graph6 (`g6`) format is used to store undirected graphs as strings.

Example 3.1. The graph $G(7,2)$ [Figure 3.1] in `g6` format is `'MhCKK@?GO`@A@Q?h?'`.

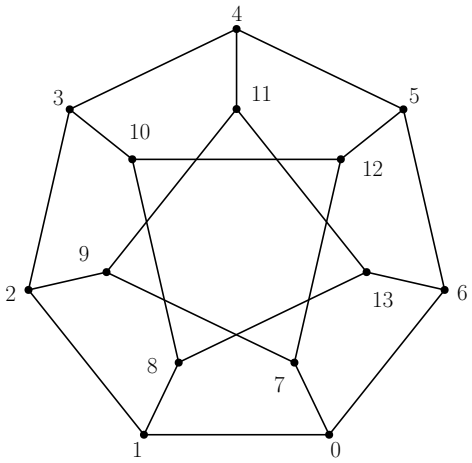


Figure 3.1: Graph $G(7,2)$.

The digraph6 (`dig6`) format is used to store directed graphs as strings. It allows antiparallel arcs, but it does not allow loops or parallel arcs.

Example 3.2. Consider the digraph D consisting on $G(7,2)$ but replacing each edge with two antiparallel arcs [Figure 3.2]. The digraph D in `dig6` format is `'MOoIA@OOIA@OOIA`?W@H?HGP@AGGP@Q?IO'`.

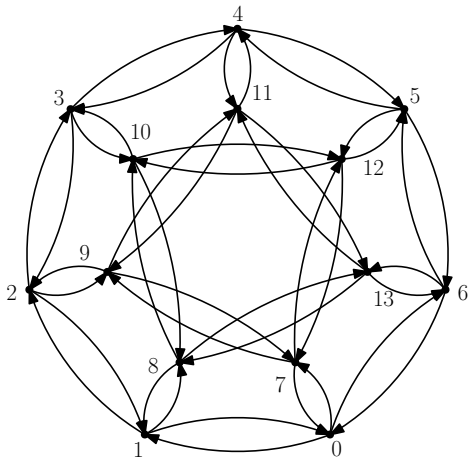


Figure 3.2: Digraph D.

Since $|C| = 3$, we are interested in 3-outregular multiorientations with loops. We are not considering loops yet, so we are interested in all the multiorientations that have at most 3 outgoing arcs for each vertex: $\delta^+(v) \leq 3 \ \forall v \in V$. Thus, in principle there could be 15 ways to orient each edge [Figure 3.3].

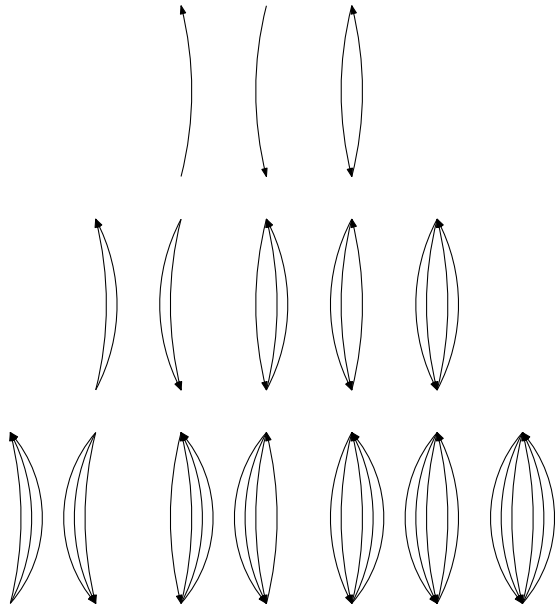


Figure 3.3: Possible multiorientations of one edge up to outdegree 3.

But parallel arcs do not change the endomorphism monoid, which in the end is what we are interested in, so we allow antiparallel arcs but not parallel arcs.

Then, each edge has only 3 possibilities: one direction, the other direction or both directions [Figure 3.4].

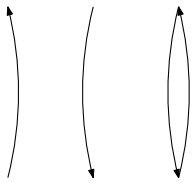


Figure 3.4: Possible multiorientations of one edge up to outdegree 1.

This would lead to $3^{21} \sim 10^{11}$ multiorientations. Nevertheless, at the end some of them represent isomorphic digraphs and we do not want to consider them. The time complexity of determining whether two finite graphs are isomorphic is unknown (graph isomorphism problem). The program `Nauty` [10] claims to be the best known program up to date to tackle this question from a practical point of view.

In conclusion, we need to compute all the multiorientations of the graph without parallel arcs and suppressing isomorphic multiorientations. The function `directg` from `Nauty` does exactly this. It requires as input the graph in format `g6`, which we obtain with SageMath, and then the function returns an output file with all the multiorientations in format `dig6`.

The resulting file contains 747197622 multiorientations. This is the total amount of multiorientations that have $G(7,2)$ as their underlying graph and that have to be analysed.

The next step is to filter these multiorientations and discard those that don't have a neutral element candidate e.

Remark 3.3. Properties of the neutral element e that we check on the code:

i. $\delta^+(e) = 3$

ii. $\delta^-(e) = 0$ (in general it can be 0 or 1, but we are in the case of no invertible elements)

iii. Every vertex is reachable from e .

iv. From e there exists at least one endomorphism to every other vertex.

Before adding the loops, we can only check properties i, ii and iii, but not iv. This is done by a straight-forward function `there_is_e_candidate_G72()`.

```
1 def everybody_reachable_from_e(d, e_candidate):
2     for v in d.vertices():
```

```
        if d.distance(e_candidate, v) > d.order():
            return False
    return True

def there_is_e_candidate_G72(d):
    for v in d.vertices():
        if d.out_degree(v) == 3 and d.in_degree(v) == 0:
            if everybody_reachable_from_e(d, v):
                return True
    return False
```

Then function `filter_the_ors_G72()` reads the file (line by line, because it is too long) and consider that the multiorientation is valid if it has a neutral element candidate.

```
def filter_the_ors_G72():
    fin = open('G72_multiors.d6', 'r')
    fout = open('G72_multiors_less_new.d6', 'w')
    line = fin.readline()
    cnt = 1
    while line:
        print(cnt, "of 747197622")
        fixed_line = line[1:]
        D = DiGraph(fixed_line, multiedges=True)
        if there_is_e_candidate_G72(D):
            fout.write(fixed_line)
        line = fin.readline()
        cnt += 1
    fin.close()
    fout.close()
```

This process takes several hours. Before filtering there are 747197622 multiorientations and after filtering there are 240088032 multiorientations, so with this simple step we get a reduction of 68% of the initial data volume.

After this first filtering, everything else is done during the execution of the main function (Figure 3.5).

Step (1) just consists on reading the file line by line. Step (2) is the following function `add_loops`. We avoid repeated loops, since they don't contribute to the endomorphism monoid of the digraph. Vertices with outdegree 0 need a loop and vertices with already outdegree 3 cannot have any loop. For vertices with outdegree 1 or 2 both possibilities are possible, since the remaining outgoing arcs could be parallel arcs to the existing ones or could be loops, so we try all the

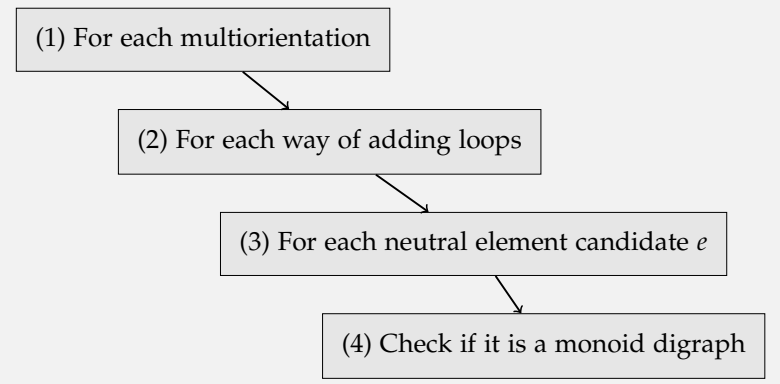


Figure 3.5: Structure of function `MAIN_approach1()`

possibilities.

```
def add_loops(D):
    vertices_outdeg_1_2 = []
    for v in D.vertices(sort=True):
        if D.out_degree(v) == 1 or D.out_degree(v) == 2:
            vertices_outdeg_1_2.append(v)
    D_essential_loops = DiGraph(D,loops=True)
    for v in D_essential_loops.vertices(sort=True):
        if D_essential_loops.out_degree(v) == 0:
            D_essential_loops.add_edge(v, v)
    done = []
    for V in powerset(vertices_outdeg_1_2):
        if V not in done:
            D_loops = D_essential_loops.copy()
            for v in V:
                D_loops.add_edge(v, v)
            yield D_loops
            for phi in D.automorphism_group().list():
                phi_of_V = im_aut(phi,V)
                if phi_of_V not in done:
                    done.append(phi_of_V)
```

Step (3) is a function `find_e_candidates_G72_loops()` that now checks all the properties that e must fulfill stated in Remark 3.3.

```
def exist_endos_from_e(d, e):
    n = d.order()
    f = [0 for _ in range(n)]
```

```
    for v in range(n):
        if v != e:
            f[e] = v
            endos_list = digraph_endos(d, e, f)
            good_endos_list = []
            for f_i in endos_list:
                if not outdeg_bad(d, f_i):
                    good_endos_list.append(f_i)
            if not good_endos_list:
                return False
    return True

def find_e_candidates_G72_loops(d):
    candidates = []
    for v in d.vertices(sort=False):
        if v not in candidates:
            if d.out_degree(v) == 3 and d.in_degree(v) == 0:
                if everybody_reachable_from_e(d, v):
                    if exist_endos_from_e(d, v):
                        candidates.append(v)
    return candidates
```

Finally, step (4) is the core of the algorithm, and is a function with the following structure (Figure 3.6).

The endomorphisms are computed by a recursive function using `yield` and `yield from` in order to compute one endomorphism at a time when needed.

```
def digraph_endos(G, e, f, v=0):
    n = G.order()
    if v == n:
        if not outdeg_bad(G,f):
            if not_invertible(G,f):
                yield f
    else:
        if v == e:
            yield from digraph_endos_1(G, e, f, v+1)
        else:
            for u in G.vertices(sort=True):
                if u >= f[v]:
                    u_is_bad = False
                    for w in G.neighbors_out(v):
                        if w < v and not G.has_edge(u, f[w]):
                            u_is_bad = True
```

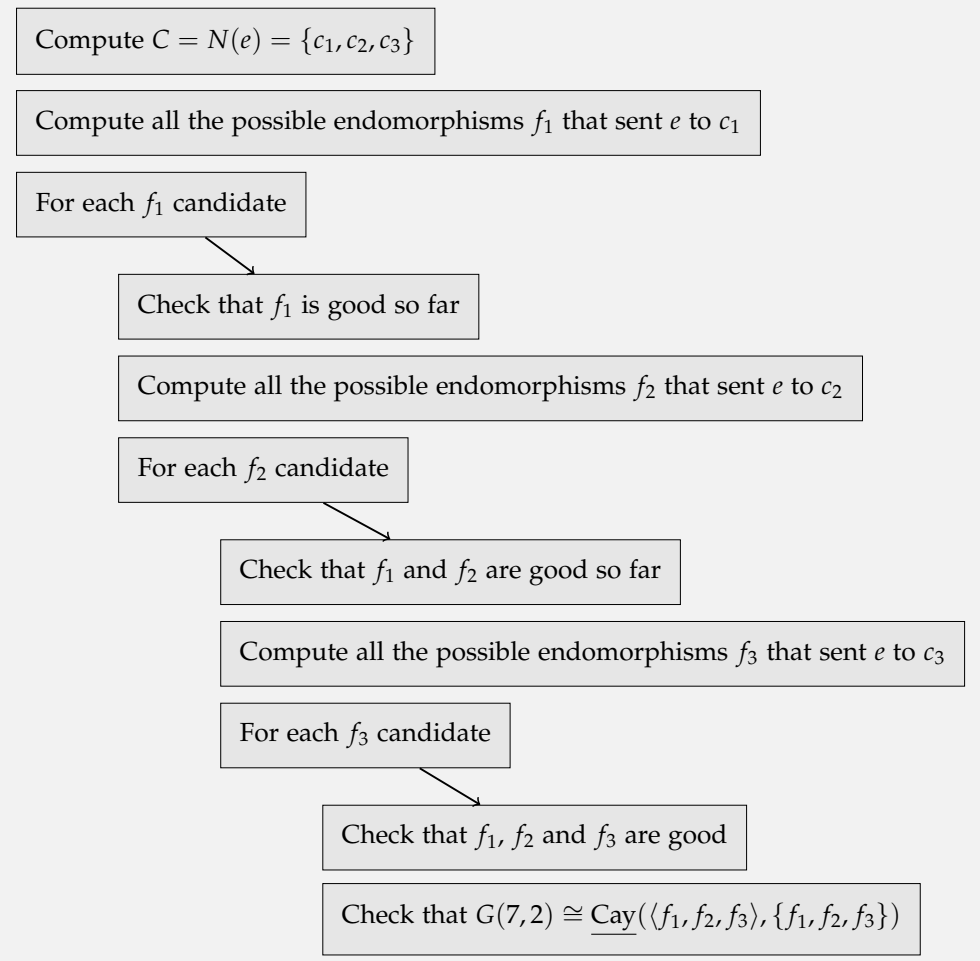


Figure 3.6: Structure of the function `check_monoid()`

```
                            break
                    if not u_is_bad:
                        for w in G.neighbors_in(v):
                            if w < v and not G.has_edge(f[w], u):
                                u_is_bad = True
                                break
                    if not u_is_bad:
                        f[v] = u
                        yield from digraph_endos_1(G, e, f, v+1)
            f[v] = 0
```

At each step, when checking if an endomorphism is good, we first check that its image does not have outgoing edges (Lemma 1.25). We also compute the closure of

the endomorphisms we have so far to see how many new endomorphisms we get by combining the existing ones, and check that we do not get more than $N = 14$ endomorphisms.

The closure of a set of endomorphisms is computed using a recursive function that stops either when the closure is completed or when it finds more than 14 endomorphisms, since we are looking for a monoid of exactly 14 elements.

```
def closure(S_original, top_s=0, m=0):
    S = S_original.copy()
    k = len(S)
    n = len(S[0])
    if top_s == 0:
        top_s = n
    if k > top_s:
        S.append("...")
        return S
    todo = []
    if m == 0:
        for i in range(k):
            todo.append(list(range(k)))
    if m != 0:
        for i in range(k-m):
            todo.append(list(range(k-m, k)))
        for i in range(k-m, k):
            todo.append(list(range(k)))
    new_k = k
    for i in range(k):
        for j in todo[i]:
            h = comp(S[i], S[j])
            new_endo = True
            for l in range(new_k):
                if h == S[l]:
                    new_endo = False
                    break
            if new_endo:
                S.append(h)
                new_k += 1
                if (new_k > top_s):
                    S.append("...")
                    return S
    if new_k == k:
        return S
    else:
```

```
37          return closure(S, top_s, new_k-k)
```

At this point of the `check_monoid()` function, after computing the closure of the endomorphisms we have so far, we also discard the endomorphisms if on their corresponding partial multiplication table every column has a repetition, since then there would not be a right neutral element. We finally compute the Cayley graph of the endomorphisms we have so far with the corresponding elements of the connection set and check that its underlying graph is isomorphic to a subgraph of $G(7,2)$. All these intermediate checks make the program faster, since most of the times they are triggered way before computing the whole monoid.

At the last step, we actually check that we have found exactly 14 endomorphisms and that $G(7,2) \cong \underline{\mathrm{Cay}}(M,C)$. When the Cayley graph $\mathrm{Cay}(M,C)$ is computed, we also associate each endomorphism to its corresponding vertex using the function `set_vertex()` which allows us to associate any object to a graph vertex. This way we are able to recover M afterwards when needed.

```
1  def cayley_graph(M, C, directed=False):
2      n = len(M)
3      order_of_C = len(C)
4      if directed:
5          G = DiGraph(loops=True)
6      if not directed:
7          G = Graph(loops=True)
8      for i in range(n):
9          G.add_vertex(i)
10         G.set_vertex(i,M[i])
11     for i in range(n):
12         for j in C:
13             x = comp(M[i], M[j])
14             for m in range(n):
15                 if M[m] == x:
16                     G.add_edge(i, m)
17                     break
18     return G
19
20 def monoidfromcayley(G):
21     return [G.get_vertex(i) for i in range(G.order())]
```

Despite all the attempts to make the program faster by discarding bad endomorphisms as soon as possible, the execution of the code does not finish all the computations.

Executed the program for 125 lines of the input file, distributed all along the file, and obtained an average computation time per line of about 1 hour and 20 minutes per line (using CPU cores of 2.0GHz), with some lines being computed in a few minutes and other lines lasting for several days. Since the file contains 240088032 lines to be analyzed, the estimated total computation time is of about 40000 years. The sample is small so this is a very rough estimation, just to grasp the order of magnitude.

3.2.2 Case B: C has exactly one involution

Now we are under the hypothesis that C has one invertible element c of order 2. Then $\langle c \rangle$ is a group, so $\mathrm{Cay}(\langle c \rangle, \{c\})$ is a regular (Corollary 1.21) and connected (Proposition 1.22) 1-outregular subdigraph of $\mathrm{Cay}(M, C)$, so it must be a cycle. Since it is of order 2, $\mathrm{Cay}(\langle c \rangle, \{c\})$ consists on two antiparallel arcs between vertex e and vertex c.

The endomorphism λ_c is an involution, so λ_c is an automorphism of the graph of order 2 that inverts the edge $\{e, c\}$. We know that $\mathrm{Aut}(G(7,2)) = D_{7\cdot 2}$ (Proposition 2.6), so λ_c has to be a reflection (Figure 3.7).

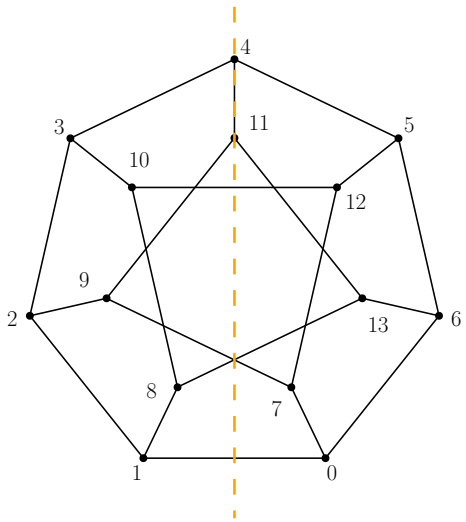


Figure 3.7: The reflection λ_c, marked with a dashed line.

The fact that $\mathrm{Aut}(G(7,2)) = D_{7\cdot 2}$ also tells us that there is no automorphism that brings the inner vertices out and viceversa, so we have two possible locations of the edge $\{e, c\}$ (Figure 3.8).

Since λ_c is an automorphism of the colored graph, orientations and colors must be preserved by this reflection. In particular, this implies that the vertices located on the symmetry axis can't have outgoing arcs leaving the axis. This is because if

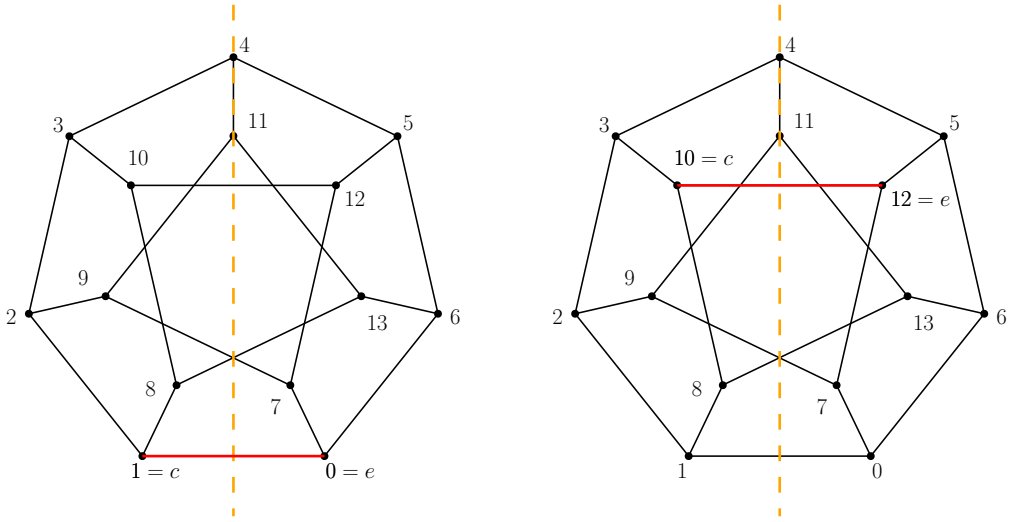


Figure 3.8: Two possible locations of the edge (e, c) when c is an involution. The reflection λ_c is marked with a dashed line.

an arc of one color leaves the axis, there must be a second symmetrical outgoing arc of the same color from the same vertex, but this cannot happen because any vertex only has one outgoing arc of each color by definition.

In conclusion, for each choice of the location of e there are some arcs that are mandatory, some arcs that are forbidden and the rest is free, meaning that can be oriented in one direction, the other or both. In the following image (Figure 3.9) we represent the two possible partial orientations that we determined. The fixed arcs are represented in red and the free edges in black.

First of all, for each partial orientation we create a file with all the possible orientations it allows, by trying all the possibilities for the free edges that lead to a symmetrical multiorientation.

For the first partial orientation we find 1248 possible symmetrical multiorientations, and for the second partial orientation we find 528 possibilities, so in total there are only 1776 multiorientations to be analyzed.

Then, the main function has a similar structure to the main function in the previous section, but now we already know where the neutral element is in each case, since we build the multiorientations from scratch (3.10).

Step (3) is essentially the same function than in the previous section, but without the intermediate checks that are specific for the case of no invertible elements.

This function does finish all the computations in about 6 hours. The result is that any of these symmetric digraphs is a monoid digraph:

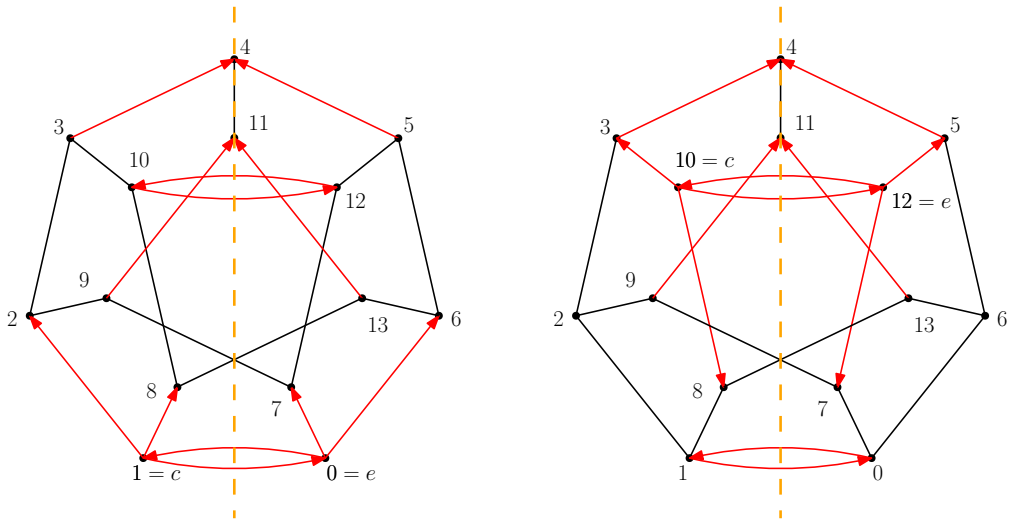


Figure 3.9: The two possible partial orientations determined by the symmetry.

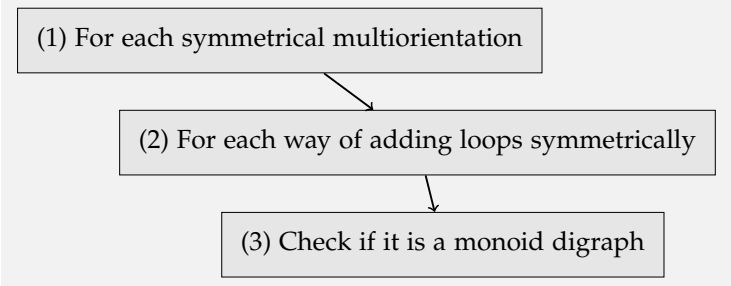


Figure 3.10: Structure of function `MAIN_approach1_INV()`

Proposition 3.4. *If* $G(7,2) \cong \underline{\text{Cay}}(M,C)$ *with* $\langle C \rangle = M$*, then* C *does not have any invertible elements.*

Proof. We know by Lemma 2.19 and Lemma 2.20 that at most C could have one invertible element and it is an involution. Since $\text{Aut}(G(7,2)) \cong D_{7\cdot 2}$ by Proposition 2.6, this involution has to be a reflection. As explained along this section, using computation one can check all the possible multiorientations with loops that are compatible with the reflection, and get that none of these possibilities correspond to a monoid digraph with $\langle C \rangle = M$. □

3.3 Approach 2: colored digraph endomorphisms

3.3.1 Case A: C has no invertible elements

The starting point is common with the first approach. But now, for every looped multiorientation, we try all the possible ways to add colors to it, in order to compute directly the endomorphisms monoid of the colored multidigraphs. As already said, the motivation to do such an expensive extra step is that if $G(7,2) \cong \underline{\text{Cay}}(M, C)$ with $\langle C \rangle = M$ then exists one of this colored multidigraphs D_{col} such that $\text{End}(D_{col})$ isomorphic to the monoid M we are looking for. So the structure of the main function for this section is the following (Figure 3.11).

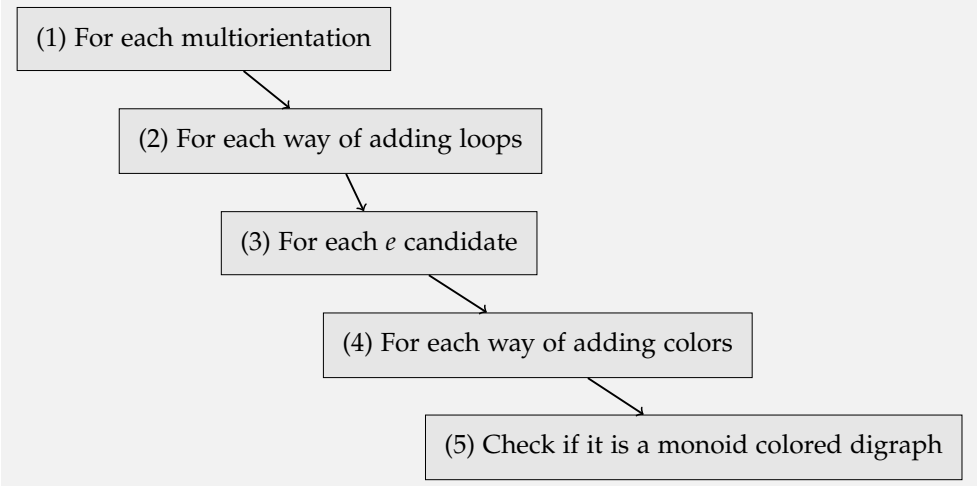


Figure 3.11: Structure of function `MAIN_approach2()`

Steps (1), (2) and (3) are exactly the same as they were in the first approach. The only difference is that then we weren't considering parallel arcs but now we have to consider them because they have different colors. So after these 3 steps, for every vertex v if $\delta^+(v) \neq 3$ we have to repeat the existing arcs in all the possible ways to get $\delta^+(v) = 3$, in order to fill the digraph until it is exactly 3-outregular.

One option is to first fill the digraph and then add colors to it. Another option is doing both things together in one single function, so that we add the repeated arcs at the very last moment before coloring them. We will be discarding bad partial colorations before finishing them, so we prefer the second option since it avoids unnecessary computations.

```
1 def change_colors_v(d, v, colors):
2     k = 3
3     edges = edges_out_v(d, v)
```

```
    d.delete_edges(edges)
    for i in range(k):
        edges[i][2] = colors[i]
    d.add_edges(edges)

def add_colors(d, e, D_with_colors=DiGraph(multiedges=True,loops=True), v=0):
    import itertools
    n = d.order()
    k = 3
    colors = list(range(k))
    if D_with_colors == DiGraph(multiedges=True,loops=True):
        D_with_colors = d.copy()
    if not cycles_good(D_with_colors, e):
        yield None
    else:
        if v == n:
            yield D_with_colors
        else:
            if v == e:
                change_colors_v(D_with_colors, v, colors)
                yield from add_colors(d, e, D_with_colors, v+1)
            else:
                edges = edges_out_v(D_with_colors, v)
                outdeg = len(edges)
                if outdeg != D_with_colors.out_degree(v):
                    print("ERROR 0 (add_colors)")
                if outdeg == 1:
                    edge = edges[0]
                    D_with_colors.add_edges([edge, edge])
                    yield from add_colors(d, e, D_with_colors, v)
                elif outdeg == 2:
                    edge1 = edges[0]
                    edge2 = edges[1]
                    D_with_colors_1 = D_with_colors.copy()
                    D_with_colors_1.add_edge(edge1)
                    yield from add_colors(d, e, D_with_colors_1, v)
                    D_with_colors_2 = D_with_colors.copy()
                    D_with_colors_2.add_edge(edge2)
                    yield from add_colors(d, e, D_with_colors_2, v)
                elif outdeg == 3:
                    parallel_edges = parallel_edges_v(D_with_colors, v)
                    if parallel_edges == [0,1,2]:
                        change_colors_v(D_with_colors, v, colors)
                        yield from add_colors(d, e, D_with_colors, v+1)
```

```
48                      elif parallel_edges == [0,1]:
49                          list_of_possible_colors_1 = [[0,1,2], [0,2,1], [1,2,0]]
50                          for possible_colors in list_of_possible_colors_1:
51                              change_colors_v(D_with_colors, v, possible_colors)
52                              yield from add_colors(d, e, D_with_colors, v+1)
53                      elif parallel_edges == [1,2]:
54                          list_of_possible_colors_2 = [[0,1,2], [1,0,2], [2,0,1]]
55                          for possible_colors in list_of_possible_colors_2:
56                              change_colors_v(D_with_colors, v, possible_colors)
57                              yield from add_colors(d, e, D_with_colors, v+1)
58                      elif parallel_edges == []:
59                          for perm_colors in itertools.permutations(colors):
60                              change_colors_v(D_with_colors, v, perm_colors)
61                              yield from add_colors(d, e, D_with_colors, v+1)
62                      else:
63                          print("ERROR 1 (add_colors)")
64                          exit()
65                  else:
66                      print("ERROR 2 (add_colors)")
67                      exit()
```

Each time the function calls itself, it checks that the partial coloration is good by checking the cycles of the subdigraphs of each color. It uses that each colored digraph is 1-outregular, and that all its connected components have shorter cycles than the component of the neutral element e and the length of their branch devides the length of the branch of the component of the neutral element e (Theorem 1.38).

Finally, step (4) of the main function is to check if the colored multidigraph is a colored monoid digraph of the graph. The function has a quite simple structure (Figure 3.12).

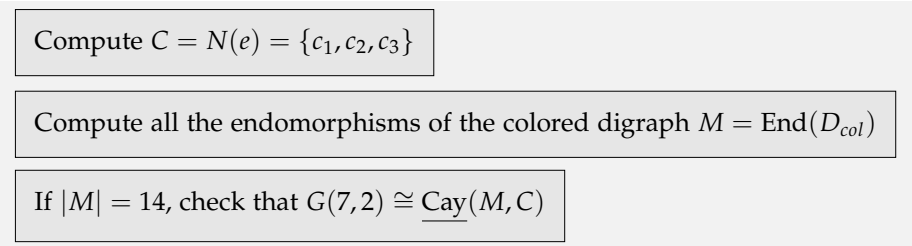


Figure 3.12: Structure of the function `check_monoid()`

The endomorphisms are computed in the same way as for looped multidigraphs, but now they are also require to respect the colors.

Again, the main function does not finish checking all the possibilities, since

computing all possible colorations is very expensive.

For this approach we only had the time to execute the program for 25 lines of the input file, and compared with the performance of the first approach on the same lines. The second approach took on average twice as long as the first approach to execute the same lines, so the estimated total computation time is of about 80000 years.

3.3.2 Case B: C has exactly one involution

In this case we follow the same procedure of the first approach, starting with the two possible symmetrical partial orientations and computing all the possible looped multidigraphs they define. Afterwards, we add colors symmetrically before checking if it is a monoid graphs.

The main function for this case is the following (3.13).

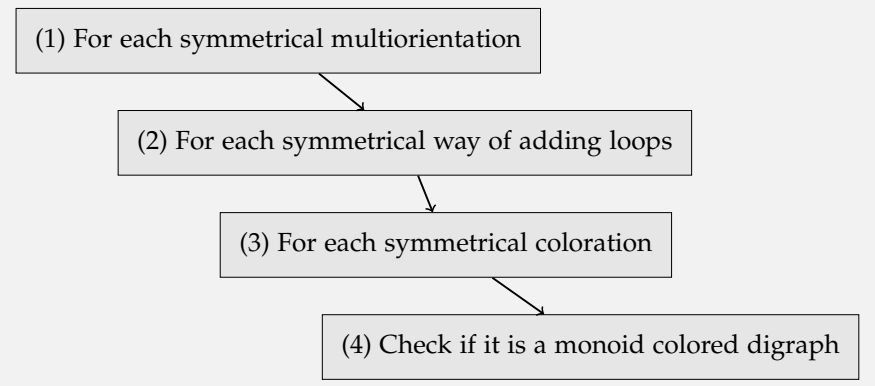


Figure 3.13: Structure of function `MAIN_approach1_INV()`

This function does finish all the computations in about 3 hours, and returns the same result that the first approach: that none of the possible symmetrical colored multiorientations of $G(7,2)$ are Cayley colored graphs of a monoid.

3.4 If C is not a generating set

For the first approach, if we drop the hypothesis $\langle C \rangle = M$ we cannot use anymore that $M \cong \langle f_1, f_2, f_3 \rangle$, where $f_j(e) = c_j \in C = N(e)$, but instead we have to compute all the 14 endomorphisms $M \cong \{f_i\}_{i=0,\dots,13} \leq \mathrm{End}(D)$, where $f_i(e) = v_i \; \forall v_i \in V$, each time. This makes it inconvenient to adapt.

For the second approach, if we drop the hypothesis $\langle C \rangle = M$ we cannot check that $M \cong \mathrm{End}(D_{col})$ but instead $M \leq \mathrm{End}(D_{col})$. This makes this approach easier to adapt to the case of C not being a generating set of M.

We adapted the algorithm of the second approach by trying all submonoids M of size 14 of $\mathrm{End}(D_{col})$, and we also adapted the filtering of the orientations demanding that not everybody is reachable from e this time.

With this small changes we were able test the case where C has an involution but without the hypothesis $\langle C \rangle = M$. This function does finish all the computations in about 3 minutes, and returns the same negative result: none of the possibilities corresponds to a monoid digraph. Therefore, we proved the following result.

Proposition 3.5. *If $G(7,2) \cong \underline{\mathrm{Cay}}(M, C)$, then C does not have any invertible elements.*

Proof. As explained along this section and chapter, using computation one can check all the possible colored multiorientations with loops that are compatible with the reflection, and get that none of these possibilities correspond to a monoid colored digraph. □

Chapter 4

Conclusions

We studied the properties of monoid graphs in general, and also the particular attributes of Generalized Petersen Graphs that are monoid graphs. We focused on the graph $G(7,2)$, and the properties that a monoid M and a connection set C need to fulfill in order for $G(7,2)$ to be isomorphic to $\underline{\mathrm{Cay}}(M,C)$.

Based on the theoretical results, we built two different algorithms to test if $G(7,2)$ is a monoid graph with monoid M and connection set C, under the hypothesis that $\langle C\rangle = M$. If C has an involution, both approaches finish computing in a relatively short period of time and give the same negative result: they don't find any monoid for $G(7,2)$. In this case, Approach 2 is faster than Approach 1. On the other hand, if C has no invertible elements, none of the algorithms finish. With the reduced sample that could be tested we estimated that, in this case, Approach 1 would be twice as fast as Approach 2, but due to the high uncertainty of the estimations we can only conclude that they are of the same order of magnitude. The results are summarized in the following table (Table 4.1).

Case	Approach	Computation time	Result
C has no invertible elements	1	$\sim 4 \cdot 10^4$ years	-
	2	$\sim 8 \cdot 10^4$ years	-
C has exactly one involution	1	6 h	False
	2	3 h	False

Table 4.1: Computation results under the hypothesis $\langle C\rangle = M$.

We adapted the second algorithm to test the case that C has an involution under the hypothesis that $\langle C\rangle \neq M$. The results are displayed in the following table (Table 4.2).

Case	Approach	Computation time	Result
C has exactly one involution	2	3 min	False

Table 4.2: Computation results under the hypothesis $\langle C \rangle \neq M$.

All together, we could draw two main conclusions:

1. We provided a computer-assisted proof that if $G(7,2)$ is a monoid graph with monoid M and connection set C, then C cannot have any invertible element.

2. We determined that, if C does not have any invertible element, the lack of symmetries makes it infeasible to check if $G(7,2)$ is a monoid graph using the computer. New theoretical results are needed to either discard this case or to set further restrictions to the multiorientations of $G(7,2)$ that are really needed to be considered.

Bibliography

[1] Laszlo Babai, *Embedding graphs in Cayley graphs*, Problèmes combinatoires et théorie des graphes, Orsay 1976, Colloq. int. CNRS No. 260 (1978), 13–15.

[2] Didier Caucal, *Cayley graphs of basic algebraic structures*, Discrete Mathematics and Theoretical Computer Science **21** (2020), no. 1, id/no. 16, 20.

[3] A. Cayley, *Desiderata and suggestions. No. 1: The theory of groups. No. 2: Graphical representation.*, American Journal of Mathematics **1** (1878), 50–52.

[4] H. S. M. Coxeter, *Self-dual configurations and regular graphs*, Bulletin of the American Mathematical Society **56** (1950), 413–455.

[5] Roberto Frucht, Jack E. Graver, and Mark E. Watkins, *The groups of the generalized Petersen graphs*, Proceedings of the Cambridge Philosophical Society **70** (1971), 211–218.

[6] Ignacio García-Marco and Kolja Knauer, *Beyond symmetry in generalized petersen graphs*, 2022.

[7] Chris Godsil and Gordon Royle, *Algebraic graph theory*, Graduate Texts in Mathematics, vol. 207, New York, NY: Springer, 2001.

[8] Kolja Knauer and Gil Puig i Surroca, *On monoid graphs*, 2021.

[9] Ulrich Knauer and Kolja Knauer, *Algebraic graph theory. Morphisms, monoids and matrices*, 2nd revised and extended edition ed., De Gruyter Studies in Mathematics, vol. 41, Berlin: De Gruyter, 2019.

[10] Brendan D. McKay and Adolfo Piperno, *Practical graph isomorphism. II.*, Journal of Symbolic Computation **60** (2014), 94–112.

[11] Roman Nedela and Martin Škoviera, *Which generalized Petersen graphs are Cayley graphs?*, Journal of Graph Theory **19** (1995), no. 1, 1–11.

[12] Gert Sabidussi, *On a class of fixed-point-free graphs*, Proceedings of the American Mathematical Society **9** (1958), 800–804.

[13] Neil J. A. Sloane, *The On-Line Encyclopedia of Integer Sequences*, Notices of the American Mathematical Society **65** (2018), no. 9, 1062–1074.

[14] Alice Steimle and William Staton, *The isomorphism classes of the generalized Petersen graphs*, Discrete Mathematics **309** (2009), no. 1, 231–237.

[15] The Sage Developers, *Sagemath, the Sage Mathematics Software System (Version 9.8)*, 2023, `https://www.sagemath.org`.

[16] Mark E. Watkins, *A theorem on Tait colorings with an application to the generalized Petersen graphs*, Proof techniques in graph theory: proceedings of the Second Ann Arbor Graph Theory Conference (1969), 171–177.

[17] Bohdan Zelinka, *Graphs of semigroups*, Časopis Pro Pěstování Matematiky **106** (1981), 407–408.