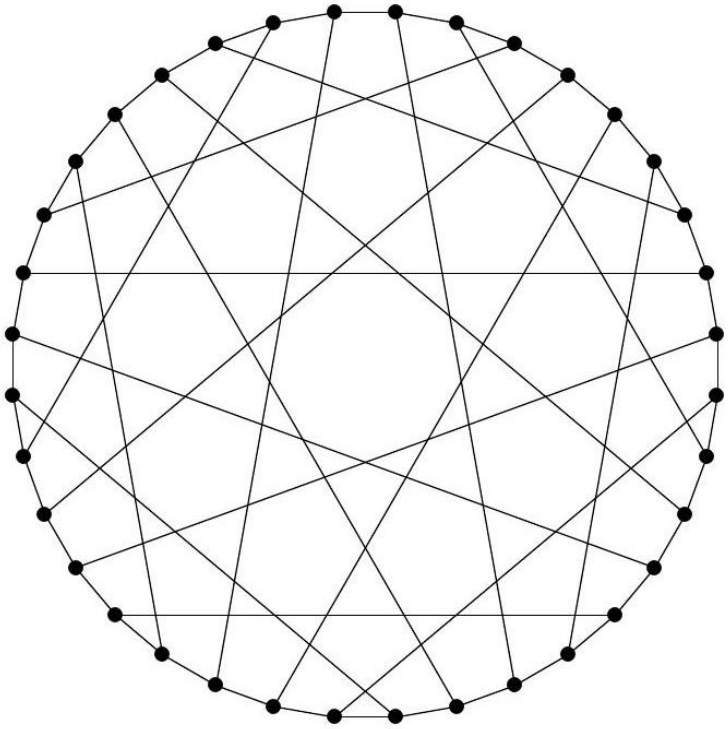
Generation of regular graphs

Markus Meringer



Diploma thesis with Prof. Dr. Laue
Chair II for Mathematics at the University of Bayreuth

The generation of complete lists of non-isomorphic regular graphs is one of the oldest problems in constructive combinatorics. Attempts were made towards the end of the last century to create complete lists of 3-regular graphs with a given number of nodes. In 1889, Jan de Vries succeeded in specifying all 3-regular graphs with 10 nodes. With the development of powerful computers, decisive progress was also made in this area.

This paper describes an algorithm for generating k-regular graphs with n nodes for a given k and n. The starting point for this was a method for constructing 3-regular graphs presented by Gunnar Brinkmann in [Bri]. In order to achieve high efficiency even for larger degrees, I also use techniques that have already proven themselves in the existing MOLGEN generator.

On the basis of these methods, a generator was developed with which even large problems can be processed in a reasonable time, such as the construction of the 4-regular graphs with 18 nodes and the 5-regular graphs with 16 nodes. It is also possible to specify a lower limit for the waist size of the generated graphs (a 3-regular graph with 36 nodes and waist size 8 is shown on the title page). Remarkable results are also achieved with this restriction. For example, all four (5,5)-cages were calculated. This problem is thus completely solved.

However, the actual intention of this work was not the "discovery of new graphs". Rather, it is integrated into the following larger framework: Regular graphs together with their automorphism groups form the input for an algorithm that constructs simple graphs for a given degree partition via the homomorphism principle. This method was implemented by Thomas Grüner as part of a diploma thesis [Grü]. In terms of range, GRADPART is already clearly superior to comparable generators in its current form. The future will show whether it will be possible to extend the program to the extent that given restrictions can be taken into account, and whether it can play an important role in interdisciplinary research projects, similar to MOLGEN.

I would especially like to thank Prof. Laue for entrusting me with a topic that aroused my interest in constructive combinatorics [Lau], and also for giving me the opportunity to develop my computer skills. Furthermore, I would like to thank Thomas Grüner for his cooperation in integrating my program part into GRADPART and for proofreading.

Markus Meringer

Contents

List of Figures

List of Tables

CHAPTER 1

Foundations

{ch1}

This chapter comprises some basic definitions and theorems about groups and operations (for more details see [?]), as well as elementary concepts from graph theory, forming the foundation for our considerations. Additionally, the lexicographical order underlying the order-preserving generation is introduced.

1.1. Groups

DEFINITION 1.1.1. Let G be a non-empty set and $\circ : G \times G \longrightarrow G$ be a mapping. The pair $(G, \circ)$ is called a group if the following conditions are satisfied:

- $\forall a, b, c \in G : \ \circ(a, \circ(b, c)) = \circ(\circ(a, b), c)$.
- $\exists e \in G : \ \circ(a, e) = a \forall a \in G$.
- $\forall a \in G \ \exists a' \in G : \ \circ\left(a, a'\right) = e$.

The element e is called the identity element of the group and is denoted by id hereafter. The above a' is called the inverse of a and is denoted by a^{-1}. The mapping $\circ$ is written as the operation of two group elements: $ab \coloneqq a \circ b \coloneqq \circ(a, b)$.

Let $(G, \circ)$ be a group and $U \subseteq G$. The pair $(U, \circ|_U)$ is called a subgroup of G if it holds that $\circ\left(u, v^{-1}\right) \in U \forall u, v \in U$. We write $U \leq G$.

EXAMPLE 1.1.2. Let $\Omega \coloneqq \{\omega_1, \omega_2, \ldots, \omega_n\}$ be a finite set with n elements. The set of all bijective mappings from Ω to Ω, together with composition, forms a group S_Ω, the symmetric group of degree n on Ω.

For $\underline{n} \coloneqq \{1, 2, \ldots, n\} \subseteq \mathbb{N}$, this group is denoted by $S_n \coloneqq S_{\underline{n}}$. The subgroups of S_n are called permutation groups of degree n, and their elements are called permutations. For permutations $\pi, \tau \in S_n$ and $i \in \underline{n}$, we have $\pi(\tau(i)) = (\pi\tau)(i)$.

DEFINITION 1.1.3. For $n \in \mathbb{N}, n > 0$, any finite sequence of natural numbers

$$\lambda := (\lambda_1, \lambda_2, \ldots, \lambda_m) \text{ with } \sum_i \lambda_i = n$$

is called a partition of n (shortly $\lambda \vDash n$). Moreover, if $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ holds, we call the partition λ canonical (shortly $\lambda \vdash n$).

REMARK 1.1.4. By a partition $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m) \vDash n$, a division of the set $\underline{n}$ into disjoint subsets is made:

$$\underline{n}_i^{\lambda} := \left\{ x \in \underline{n} \mid \sum_{\nu=1}^{i-1} \lambda_\nu < x \leq \sum_{\nu=1}^{i} \lambda_\nu \right\}, \quad i = 1, \ldots, m$$

where $\underline{n}_i^{\lambda}$ contains exactly λ_i elements. It holds:

$$\underline{n} = \bigcup_{i=1}^{m} \underline{n}_i^{\lambda}.$$

We define the Young subgroup of S_n associated with λ as:

$$S_\lambda := \prod_{i=1}^{m} S_{\underline{n}_i^{\lambda}}.$$

1.2. Operations

DEFINITION 1.2.1. Let G be a finite group and Ω a non-empty finite set. A mapping

$$\varphi : \Omega \times G \longrightarrow \Omega, \quad (\omega, g) \longmapsto \omega^g := \varphi(\omega, g)$$

is called an operation of G on Ω if φ is compatible with the operation in G and if id fixes the elements of Ω. So, the following conditions must hold:

- $\forall \omega \in \Omega,\ \forall g, g' \in G :\ \omega^{gg'} = (\omega^g)^{g'}$.
- $\forall \omega \in \Omega :\ \omega^{id} = \omega$.

If G operates on Ω, this operation is denoted by ${}_G\Omega$. The operation is called *transitive* if:

$$\forall \omega, \omega' \in \Omega\ \exists g \in G :\ \omega^g = \omega'$$

EXAMPLE 1.2.2. A very simple example is provided by the operation of S_n on $\underline{n}$. If we define $x^\pi := \pi^{-1}(x)$, then the above conditions are fulfilled for $\pi, \tau \in S_n$, $x \in \underline{n}$, since $id^{-1}(x) = x$ and

$$x^{\pi\tau} = (\pi\tau)^{-1}(x) = \left(\tau^{-1}\pi^{-1}\right)(x) = \tau^{-1}\left(\pi^{-1}(x)\right) = \tau^{-1}\left(x^\pi\right) = \left(x^\pi\right)^\tau.$$

{defOperation}

DEFINITION 1.2.3. Let ${}_G\Omega$ be an operation, $U \subseteq G$, $\omega \in \Omega$, and $\Delta \subseteq \Omega$.

i) $C_G(\Delta) := \{g \in G \mid \delta^g = \delta \ \forall \delta \in \Delta\}$ is called the centralizer or pointwise stabilizer of Δ in G with respect to ${}_G\Omega$. If $\Delta = \{\delta\}$, it is also written as $C_G(\delta)$.

ii) $N_G(\Delta) := \{g \in G \mid \delta^g \in \Delta \ \forall \delta \in \Delta\}$ is called the normalizer or setwise stabilizer of Δ in G with respect to ${}_G\Omega$.

iii) $\mathrm{Fix}_\Omega(U) := \{\omega \in \Omega \mid \omega^u = \omega \ \forall u \in U\}$ is called the fixed point set of $U \subseteq G$. An element of $\mathrm{Fix}_\Omega(U)$ is called a fixed point of U.

iv) $\omega^G := \{\omega^g \in \Omega \mid g \in G\}$ is called the orbit of ω under ${}_G\Omega$.

v) $G\backslash\backslash\Omega := \left\{\omega^G \mid \omega \in \Omega\right\}$ is the set of orbits of the operation ${}_G\Omega$.

DEFINITION 1.2.4. Let G be a group, Ω a set, and ${}_G\Omega$ an operation. A set $T \subseteq \Omega$ is called a complete set of representatives or transversal of the orbits of this operation if it satisfies:

$$|B \cap T| = 1 \quad \forall B \in G\backslash\backslash\Omega.$$

So, a transversal T contains exactly one element from each orbit. The set of all transversals is denoted by $\mathcal{T}(G\backslash\backslash\Omega)$.

The concept of orbit from Definition 1.2.3 can also be defined via an equivalence relation. This has the advantage that elementary properties of orbits can be recognized without calculation.

{lmOperation}

LEMMA 1.2.5. *Let G be a group, Ω a set, and ${}_G\Omega$ an operation. Define the following relation on Ω:*

$$\omega \sim \omega' \quad \Leftrightarrow \quad \exists g \in G : \omega^g = \omega'.$$

Then $\sim$ is an equivalence relation, and the orbits of the operation ${}_G\Omega$ correspond to the equivalence classes of $\sim$.

PROOF. The properties of an equivalence relation are straightforward to verify: $\omega \sim \omega$ since $\omega^{id} = \omega$; $\omega \sim \omega' \Rightarrow \exists g \in G : \omega^g = \omega' \Rightarrow \omega'^{g^{-1}} = (\omega^g)^{g^{-1}} = \omega^{gg^{-1}} = \omega \Rightarrow \omega' \sim \omega$;

$\omega \sim \omega', \omega' \sim \omega'' \Rightarrow \exists g, g' \in G : \omega^g = \omega', \omega'^{g'} = \omega'' \Rightarrow \omega^{gg'} = (\omega^g)^{g'} = \omega'^{g'} = \omega'' \Rightarrow \omega \sim \omega''$ for all $\omega, \omega', \omega'' \in \Omega$. The relationship between orbits and equivalence classes follows directly from the definition of $\sim$. □

{thmOrbits}

THEOREM 1.2.6. *Let G be a group, Ω a set, and ${}_G\Omega$ an operation. Then:*

- *Any two orbits of ${}_G\Omega$ are either equal or disjoint:*
$$\forall \omega, \omega' \in \Omega : \quad \omega^G \cap \omega'^G \neq \varnothing \iff \omega' \in \omega^G.$$
- *Ω is the disjoint union of the orbits of ${}_G\Omega$:*
$$\forall T \in \mathcal{T}(G\backslash\backslash\Omega) : \quad \Omega = \mathring{\bigcup_{\omega \in T}} \omega^G.$$

PROOF. The statements follow from Lemma 1.2.5 and the properties of equivalence classes. □

{exCoset}

EXAMPLE AND DEFINITION 1.2.7. Let G be a group, $U \leq G$ a subgroup. Then, the mapping
$$G \times U \longrightarrow G, \quad (g, u) \longmapsto g^u := gu$$
defines an operation of U on G. For $g \in G$, the orbit
$$gU := g^U = \{g^u \mid u \in U\} = \{gu \mid u \in U\}$$
is called the left coset of U in G. The set of all left cosets of U in G is denoted by G/U. According to Theorem refthmOrbits, any two left cosets are either equal or disjoint, and G is the disjoint union of the left cosets of U in G.

DEFINITION 1.2.8. Let G be a group, Ω a set, and ${}_G\Omega$ an operation. Further, let $T \in \mathcal{T}(G\backslash\backslash\Omega)$ be a transversal of the orbits of this operation. A mapping $\rho : \Omega \longrightarrow G$ is called a fusing mapping to T if
$$\forall \omega \in \Omega : \quad \omega^{\rho(\omega)} \in T.$$
Thus, the fusing element $\rho(\omega)$ maps ω to the corresponding orbit representative.

REMARK 1.2.9. A fusing mapping ρ becomes particularly important when comparing two orbits ω^G and ω'^G. Instead of testing whether ω' is in ω^G, one only needs to determine the two elements $\omega^{\rho(\omega)}$ and $\omega'^{\rho(\omega')}$. Then,
$$\omega^G = \omega'^G \iff \omega^{\rho(\omega)} = \omega'^{\rho(\omega')}.$$

{rkActionSubsets}

REMARK 1.2.10. Let G be a group, X a set, and ${}_GX$ an operation. Then, we want to consider the following induced operations:

- G operates on the set $\binom{X}{p}$ of p-element subsets of $X, 1 < p < |X|$, as follows:
$$\binom{X}{p} \times G \longrightarrow \binom{X}{p}, \quad \{x_1, \ldots, x_p\}^g := \{x_1{}^g, \ldots, x_p{}^g\}$$

- Let Y be a set. Then, an operation of G on Y^X, the set of functions γ from X to Y, can be defined as follows:

$$Y^X \times G \longrightarrow Y^X, \quad (\gamma, g) \longmapsto \gamma^g \text{ with } \left(\gamma^g\right)(x) \coloneqq \gamma\left(x^{g^{-1}}\right) \ \forall x \in X$$

We want to verify the properties of an operation for the case ${}_G\left(Y^X\right)$:

$$\left(\gamma^{gg'}\right)(x) = \gamma\left(x^{(gg')^{-1}}\right) = \gamma\left(x^{g'^{-1}g^{-1}}\right) = \gamma\left(\left(x^{g'^{-1}}\right)^{g^{-1}}\right) = \left(\gamma^g\right)\left(x^{g'^{-1}}\right) =$$

$\left(\left(\gamma^g\right)^{g'}\right)(x)$ and $\left(\gamma^{id}\right)(x) = \gamma\left(x^{id}\right) = \gamma(x) \forall x \in X, \forall \gamma \in Y^X, \forall g, g' \in G$.

Definition 1.2.11. Let $n \in \mathbb{N}, n > 1$. We define

$$X \coloneqq \binom{\underline{n}}{2}$$

as the set of 2-element subsets of $\underline{n}$. Then, the elements of

$$\mathcal{G}_n \coloneqq \{0,1\}^X$$

are called labeled simple graphs with n nodes. For $\Gamma \in \mathcal{G}_n$, we call the set

$$E \coloneqq \{e \in X \mid \Gamma(e) = 1\}$$

the edge set of Γ. Two nodes $i, j \in \underline{n}, i \neq j$ are adjacent if $\Gamma(\{i,j\}) = 1$. The degree of node i is defined as the number of nodes adjacent to i:

$$\deg(i) \coloneqq \deg_\Gamma(i) \coloneqq \sum_{j \in \underline{n}, j \neq i} \Gamma(\{i,j\}).$$

If all nodes have the same degree k, we refer to it as a labeled k-regular graph with n nodes. The set of these graphs is denoted by

$$\mathcal{R}_{n,k} \coloneqq \left\{\Gamma \in \mathcal{G}_n \mid \deg_\Gamma(i) = k \forall i \in \underline{n}\right\} \subseteq \mathcal{G}_n.$$

{defActionOnGraphs}

Definition 1.2.12. Following Remark 1.2.10, we define the following operation of S_n on $\mathcal{G}_n$:

$$\mathcal{G}_n \times S_n \longrightarrow \mathcal{G}_n, \quad (\Gamma, \pi) \longmapsto \Gamma^\pi \quad \text{with}$$
$$\Gamma^\pi(\{i,j\}) \coloneqq \Gamma(\{\pi(i), \pi(j)\}) \quad \forall \{i,j\} \in X.$$

Two labeled simple graphs $\Gamma, \Gamma' \in \mathcal{G}_n$ are isomorphic if they belong to the same orbit under this operation. The elements of $S_n \backslash\backslash \mathcal{G}_n$ are called simple graphs with n nodes.

With the described operation, regularity is preserved. Using the same mapping, we can also define an operation of S_n on $\mathcal{R}_{n,k}$. The elements of $S_n \backslash\backslash \mathcal{R}_{n,k}$ are called k-regular graphs with n nodes.

REMARK 1.2.13. To a labeled simple graph $\Gamma \in \mathcal{G}_n$ corresponds its adjacency matrix $A = \left(a_{i,j}\right)_{1 \le i,j \le n}$ with

$$a_{i,j} := \begin{cases} \Gamma(\{i,j\}), & \text{if } i \neq j, \\ 0, & \text{otherwise.} \end{cases}$$

The adjacency matrix A is thus a symmetric $n \times n$ matrix with entries zero or one and zeros on the diagonal. Conversely, for every such matrix A, there exists a labeled simple graph Γ, such that A is the adjacency matrix of Γ. We can identify the set of these matrices with $\mathcal{G}_n$.

The adjacency matrix will play a central role in the following considerations as a data structure for labeled graphs. In particular, we will exploit the symmetry of these matrices as much as possible without explicitly pointing it out each time. For example, when comparing two adjacency matrices, we only need to consider the upper right halves. In particular, we will compare A and A^π for $\pi \in S_n$, where A^π is the adjacency matrix of Γ^π :

$$A^\pi = \left(a_{\pi(i),\pi(j)}\right)_{1 \le i,j \le n}.$$

We obtain A^π by simultaneously permuting the rows and columns of A.

NOTATION 1.2.14. Since this work exclusively deals with simple graphs, we can omit the adverb "simple" in the following. Therefore, $S_n \backslash\backslash \mathcal{G}_n$ denotes the set of graphs and $\mathcal{G}_n$ denotes the set of labeled graphs with n points.

A labeled graph $\Gamma \in \mathcal{G}_n$ is uniquely determined by the set of its edges $E = \{e_1, \ldots, e_t\}$. We will use this property to describe Γ by its edge set: $\Gamma = \{e_1, \ldots, e_t\}$. For $\pi \in S_n$, $\Gamma^\pi = \{{e_1}^\pi, \ldots, {e_t}^\pi\}$.

For two labeled graphs Γ and Γ', $\Gamma \subseteq \Gamma'$ means that $E \subseteq E'$, where E, E' denote the respective edge sets.

When we use the notation $e = \{v, w\} \in E$ for an edge e (with "round brackets"), we assume that $v < w$.

1.3. Graph Theory

DEFINITION 1.3.1. Let $\Gamma \in \mathcal{G}_n$ be a labeled graph. The edges $e_1, e_2, \ldots, e_q$ of Γ form an edge sequence if, for $i = 2, 3, \ldots, q-1$, edge e_i shares one of its nodes with e_{i-1} and the other with e_{i+1}. A sequence of edges without repetition is called a trail. A path is a trail that does not repeat any node. We describe a path connecting nodes v and w by listing the nodes it passes through: $W = \left(v, v_1, v_2, \ldots, v_{q-1}, w\right)$. We

refer to nodes v and w as endpoints of W. The number q of edges in the path W is called its length $l(W)$: $l(W) = q$. The length of the shortest path connecting v and w is called the distance from v to w:

$$\mathrm{dist}_\Gamma(v,w) := \min\{l(W) \mid W \text{ is a path connecting } v \text{ and } w\}$$

if such a path exists. Otherwise, we set $\mathrm{dist}_\Gamma(v,w) = \infty$, and specifically for $v = w$: $\mathrm{dist}_\Gamma(v,v) = 0$. Furthermore, for an edge $\left\{u,u'\right\}$, we define the distance from v to $\left\{u,u'\right\}$ as

$$\mathrm{dist}_\Gamma\left(v,\left\{u,u'\right\}\right) := \min\left\{\mathrm{dist}_\Gamma(v,u), \mathrm{dist}_\Gamma\left(v,u'\right)\right\}.$$

We call Γ connected if every pair of nodes is connected by a path.

REMARK 1.3.2. The relation defined for pairs of nodes, connected by a path, is an equivalence relation. The equivalence classes for this relation are called connected components. Thus, Γ is connected if and only if there is only one connected component. A node with degree 0 is called a trivial connected component.

{defRegularLabelled}

DEFINITION 1.3.3. The set of connected k-regular labeled graphs with n nodes is called

$$\mathcal{R}^*_{n,k} := \left\{\Gamma \in \mathcal{R}_{n,k} \mid \Gamma \text{ is connected } \right\}.$$

We denote by $\mathcal{G}^*_n$ the labeled graphs $\Gamma \in \mathcal{G}_n$ that have exactly one non-trivial connected component.

DEFINITION 1.3.4. If in $\Gamma \in \mathcal{G}_n$ the two endpoints v and w of a path $W = \left(v, v_1, v_2, \ldots, v_{q-1}, w\right)$ are connected by an edge $e = \{v,w\}$, then $K := W \cup \{e\}$ forms a cycle of length $l(K) = q + 1$. If Γ has no cycles of smaller length, it is called a girth cycle. The length of a girth cycle is referred to as the girth of the graph:

$$\mathrm{girth}(\Gamma) := \min\{l(K) \mid K \text{ is a cycle in } \Gamma\}.$$

If Γ has at least one cycle, $\mathrm{girth}(\Gamma)$ is well-defined as the minimum of a finite set. Otherwise, we set $\mathrm{girth}(\Gamma) := \infty$.

DEFINITION 1.3.5. A connected labeled graph that does not contain any cycles is called a tree. Nodes with degree 1 are called leaves, and all others are called inner nodes. If a node is designated as the root, we also refer to it as a rooted tree. We will use these terms for cycle-free $\Gamma \in \mathcal{G}^*_n$ as well, where the root is always a node with degree greater than 0.

REMARK 1.3.6. Let $\Gamma \in \mathcal{G}^*_n$ be a rooted tree with root 1, and v_0 be a node with $\mathrm{dist}_\Gamma(v_0, 1) = d$. If $0 < d < \infty$, then there exists exactly one neighbor v of v_0 with $\mathrm{dist}_\Gamma(v,1) = d - 1$. This node is called the

parent of v_0. For all other neighbors w of v_0, $\mathrm{dist}_\Gamma(w,1) = d+1$. These nodes are the children of v_0. We denote the set of children of v_0 by $\mathrm{succ}_1(v_0)$. Furthermore, we define recursively for $i = 1,\dots,n-1$:

$$\mathrm{succ}_{i+1}(v_0) := \bigcup_{u\in \mathrm{succ}_i(v_0)} \mathrm{succ}_1(u).$$

Then we call the elements of $\mathrm{succ}(v_0) := \bigcup_{i=1}^{n} \mathrm{succ}_i(v_0)$ the successors of v_0.

DEFINITION 1.3.7. Let $n,k,t \in \mathbb{N}, n > 1, k < n, t \geq 3$. We define:

$$\begin{aligned}
\mathcal{G}_{n,k} &:= \left\{\Gamma \in \mathcal{G}_n \mid \deg_\Gamma(i) \leq k \text{ for all } i \in \underline{n}\right\},\\
\mathcal{G}^*_{n,k} &:= \left\{\Gamma \in \mathcal{G}^*_n \mid \deg_\Gamma(i) \leq k \text{ for all } i \in \underline{n}\right\},\\
\mathcal{R}_{n,k,t} &:= \left\{\Gamma \in \mathcal{R}_{n,k} \mid \mathrm{girth}(\Gamma) \geq t\right\},\\
\mathcal{R}^*_{n,k,t} &:= \left\{\Gamma \in \mathcal{R}^*_{n,k} \mid \mathrm{girth}(\Gamma) \geq t\right\}.
\end{aligned}$$

{rkInvariants}

REMARK 1.3.8. Girth and the number of connected components are invariants under the operation defined in Definition 1.2.12. The goal of this work will be to determine the following sets for given n,k,t:

- $S_n\backslash\backslash\mathcal{R}_{n,k}$, the k-regular graphs with n nodes,
- $S_n\backslash\backslash\mathcal{R}^*_{n,k}$, the connected k-regular graphs with n nodes, and
- $S_n\backslash\backslash\mathcal{R}^*_{n,k,t}$, the connected k-regular graphs with n nodes and girth at least t.

We want to provide a complete representative system for each of these. The following describes how the representatives are selected.

1.4. Lexicographic Order

efLexicographicOrder}

DEFINITION 1.4.1. The set $X = \binom{n}{2}$ can be ordered as follows: Let $e_1 = (v_1,w_1), e_2 = (v_2,w_2) \in X$, then

$$e_1 < e_2 \quad :\Longleftrightarrow\quad v_1 < v_2 \vee (v_1 = v_2 \wedge w_1 < w_2).$$

$\mathcal{G}_n$ is also lexicographically ordered. For two numbered graphs $\Gamma,\Gamma' \in \mathcal{G}_n$ with $\Gamma = \{e_1,\dots,e_t\}$ and $\Gamma' = \left\{e'_1,\dots,e'_{t'}\right\}$ satisfying $e_1 < \dots < e_t, e'_1 < \dots < e'_{t'}$, then

$$\begin{aligned}
\Gamma < \Gamma' \quad :\Longleftrightarrow\quad & \left(\exists i \leq \min\left\{t,t'\right\} : e_j = e'_j \ \forall j < i \wedge e_i < e'_i\right)\\
& \vee \left(t < t' \wedge e_j = e'_j \ \forall j \leq t\right).
\end{aligned}$$

{defOrderAdjacency}

DEFINITION 1.4.2. On 0/1-vectors of length l, we define the order $\succ$ as follows:

$$(a_1,\dots,a_l) \succ \left(a'_1,\dots,a'_l\right) \quad :\Longleftrightarrow\quad \exists i \leq l : a_j = a'_j \forall j < i \wedge a_i > a'_i$$

Two numbered graphs $\Gamma, \Gamma' \in \mathcal{G}_n$ can also be compared by concatenating the entries of their adjacency matrices A and A' row-wise and considering them as $0/1$-vectors of length n^2:

$$A \succ A' \quad :\Longleftrightarrow (a_{11}, \ldots, a_{1n}, \ldots\ldots, a_{n1}, \ldots, a_{nn}) \succ \left(a'_{11}, \ldots, a'_{1n}, \ldots\ldots, a'_{n1}, \ldots, a'_{nn}\right)$$

REMARK 1.4.3. At first glance, Definition 1.4.2 may seem redundant since it also defines a lexicographic order. However, in practice, the adjacency matrix serves as the primary data structure for numbered graphs when implementing algorithms on computers. Therefore, the order defined in Definition 1.4.2 is used for algorithm development. For theoretical purposes, such as formulating theorems and proofs, Definition 1.4.1 has been found to be more elegant. The following lemma provides a statement regarding the equivalence of the two orders.

{lmOrderComparison}

LEMMA 1.4.4. *If two numbered graphs $\Gamma, \Gamma' \in \mathcal{G}_n$ have the same number of edges, then for the adjacency matrices A of Γ and A' of Γ':*

$$A \succ A' \Longleftrightarrow \Gamma < \Gamma'.$$

PROOF. "$\Rightarrow$": Let (i_0, j_0) be the first position where A and A' differ, i.e., $a_{i_0 j_0} = 1 > 0 = a'_{i_0 j_0}$. Then $i_0 < j_0$. All edges $e_j, e'_j < (i_0, j_0)$ in Γ and Γ', respectively, originate from ones before position (i_0, j_0) in A and A'. Let m be the number of such edges, then $e_j = e'_j \forall\, j \leq m$. For $e_{m+1} = (i_0, j_0)$, $e_{m+1} \in \Gamma$ but $e_{m+1} \notin \Gamma'$. Since both graphs have the same number of edges, Γ' must have an edge $e'_{m+1} > e_{m+1}$. Thus, for Γ, we have edges $e_1 < \ldots < e_m < e_{m+1} < \ldots$, and for Γ', we have edges $e_1 < \ldots < e_m < e'_{m+1} < \ldots$, hence $\Gamma < \Gamma'$.

"$\Leftarrow$": Since Γ and Γ' have the same number of edges, $\exists m : e_i = e'_i \forall\, i < m, e_m < e'_m$. Let $e_m = (i_0, j_0)$. The edges e_j, e'_j with $j \geq m$ do not introduce ones before position (i_0, j_0) in A and A'. Before position (i_0, j_0), A and A' are equal. Furthermore, $a_{i_0 j_0} = 1 > 0 = a'_{i_0 j_0}$, thus $A \succ A'$. □

REMARK 1.4.5. The statement of Lemma 1.4.4 becomes false when different numbers of edges are present. An example is provided by two numbered graphs with 3 nodes:

$$\Gamma = \{(1,2)\}, A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and

$$\Gamma' = \{(1,2),(1,3)\}, A' = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Here, $\Gamma \prec \Gamma'$ and $A \prec A'$. In all cases where we need to compare graphs lexicographically, they will have the same number of edges.

{defCanonical}

DEFINITION 1.4.6. Let $\Gamma \in \mathcal{G}_n$. We call Γ canonical (with respect to the lexicographic order "$<$") if Γ is minimal in its orbit under S_n:

$$\Gamma \text{ canonical} \quad \Leftrightarrow \quad \forall \pi \in S_n : \Gamma \leq \Gamma^\pi.$$

According to Lemma 1.4.4, this is also equivalent to $A \geq A^\pi$ for all $\pi \in S_n$, where A is the adjacency matrix of Γ. The set of canonical orbit representatives

$$\mathrm{rep}_<(S_n \backslash\backslash \mathcal{G}_n) := \{\Gamma \in \mathcal{G}_n \mid \forall \pi \in S_n : \Gamma \leq \Gamma^\pi\}$$

is a canonical representative system of $S_n \backslash\backslash \mathcal{G}_n$. We will determine the sets from Remark 1.3.8 by specifying the canonical representative system (with respect to the lexicographic order "$<$") for each:

$$\mathrm{rep}_<(S_n \backslash\backslash \mathcal{R}_{n,k}) := \{\Gamma \in \mathcal{R}_{n,k} \mid \forall \pi \in S_n : \Gamma \leq \Gamma^\pi\}$$
$$\mathrm{rep}_<(S_n \backslash\backslash \mathcal{R}^*_{n,k}) := \{\Gamma \in \mathcal{R}^*_{n,k} \mid \forall \pi \in S_n : \Gamma \leq \Gamma^\pi\}$$
$$\mathrm{rep}_<(S_n \backslash\backslash \mathcal{R}^*_{n,k,t}) := \{\Gamma \in \mathcal{R}^*_{n,k,t} \mid \forall \pi \in S_n : \Gamma \leq \Gamma^\pi\}$$

CHAPTER 2

Existence Criteria for Regular Graphs

{ch2}

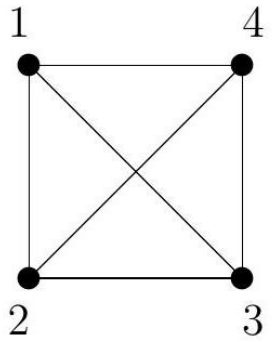
In this chapter, a statement about the existence of k-regular graphs with n nodes is made, and the trivial cases are briefly discussed. In Section 2.2, a necessary condition for the existence of a k-regular graph with n nodes and a given girth is developed. The insights gained will also be applied in Chapter 3. The concept of (k,t)-cage is introduced, and an overview of known cages is given.

2.1. Regular Graphs

REMARK 2.1.1. Let $n,k \in \mathbb{N}, n \geq 2, \Gamma \in \mathcal{R}_{n,k}$. The number of edges of Γ is $|\Gamma| = \frac{nk}{2} \in \mathbb{N}$. If both n and k are odd, then $\mathcal{R}_{n,k} = \varnothing$.

If $k = 0$, we refer to the graph with n nodes as the empty graph. For $k = 1$, there exists $\Gamma \in \mathcal{R}_{n,1}$ if n is even. Γ then has no cycles and is not connected if $n > 2$. For even n, $\left|S_n \backslash\backslash \mathcal{R}_{n,1}\right| = 1$. For $k = 2, n > 2$, $\Gamma \in \mathcal{R}^*_{n,2}$ is a cycle of length n. $\left|S_n \backslash\backslash \mathcal{R}^*_{1,n}\right| = 1$. For $k = 3$, we also call 3-regular graphs cubic graphs.

DEFINITION 2.1.2. Let $k \geq 1, n = k + 1$. The k-regular numbered graph in which every two nodes are connected by an edge is called a complete graph $K_n := \binom{n}{n-1}$ (see Fig. 2.1). There is exactly one complete graph for each $n \geq 2$: $\left|\mathcal{R}_{n,n-1}\right| = \left|S_n \backslash\backslash \mathcal{R}_{n,n-1}\right| = 1$.



figK4

FIGURE 2.1. The graph K_4.

2.2. Regular Graphs with Given Girth

DEFINITION 2.2.1. Let $k, t \in \mathbb{N}, k \geq 2, t \geq 3$. We define

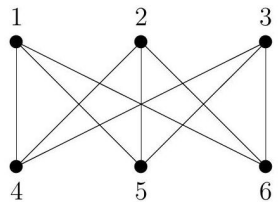
$$f(k,t) := \min\left\{n \in \mathbb{N} \mid \mathcal{R}_{n,k,t} \neq \emptyset\right\}.$$

REMARK 2.2.2. It holds that $f(2,t) = t$ for $t \geq 3$. For $t = 3$, $f(k,3) = k+1$ since $K_{k+1} \in \mathcal{R}_{k+1,k,3} \neq \emptyset$. For $t = 4$, $f(k,4) = 2k$, because the bipartite numbered graph B_{2k} with $2k$ nodes has a girth of 4: $B_{2k} := \{\{i,j\} \mid 1 \leq i \leq k \wedge k < j \leq 2k\} \in \mathcal{R}_{2k,k,4} \neq \emptyset$ (cf. Fig. 2.2).

The justification for the minimality of the number of nodes in both cases is provided by Theorem 2.2.4. For $t > 4$, the values of $f(k,t)$ are not easy to determine and are still unknown in almost all cases (see Table 2.2 and Remark 2.2.8).

•2.2.1: rg: Labels

•2.2.1



figB6

FIGURE 2.2. The graph B_6.

DEFINITION 2.2.3. Let $k \geq 2, t \geq 3$, and $n_0 = f(k,t)$. Then the elements of $S_{n_0}\backslash\backslash\mathcal{R}_{n_0,k,t}(k,t)$ are called (k,t)-cages. More generally, for $n \geq n_0$, the elements of $S_n\backslash\backslash\mathcal{R}_{n,k,t}(k,t)$ are called (k,t)-graphs.

{thmCage}

An lower bound for $f(k,t)$ is provided by he following theorem:

THEOREM 2.2.4. *For $k, t \in \mathbb{N}, k \geq 2, t \geq 3$, let*

$$f_0(k,t) := \begin{cases} 1 + k\sum_{i=1}^{\frac{t-1}{2}}(k-1)^{i-1}, & \text{if } t \text{ is odd} \\ 2\sum_{i=1}^{\frac{t}{2}}(k-1)^{i-1}, & \text{otherwise.} \end{cases}$$

Then $f(k,t) \geq f_0(k,t)$.

PROOF. Let $\Gamma \in \mathcal{R}_{n,k,t}$.

i) For $t = 2d+1$ odd. We define for $i = 0, \ldots, d$ disjoint sets

$$niv(i) := \mathrm{niv}_{\Gamma,1}(i) := \{v \in \underline{n} \mid \mathrm{dist}(1,v) = i\}.$$

For $1 \leq i \leq d$, it holds that every node $w \in \mathrm{niv}(i)$ has exactly one neighbor in $\mathrm{niv}(i-1)(*)$. If w were adjacent to two nodes

$v_1, v_2 \in \mathrm{niv}(i-1)$, then Γ would have a cycle of length

$$\mathrm{dist}(1, v_1) + \mathrm{dist}(v_1, w) + \mathrm{dist}(w, v_2) + \mathrm{dist}(v_2, 1) = 2i \leq 2d < t$$

which is a contradiction.

Furthermore, for $1 \leq i < d$, every node $w \in \mathrm{niv}(i)$ has exactly $k-1$ neighbors in $\mathrm{niv}(i+1)(*)$. If w were adjacent to a node $w' \in \mathrm{niv}(i)$, then Γ would have a cycle of length

$$\mathrm{dist}(1, w) + \mathrm{dist}\left(w, w'\right) + \mathrm{dist}\left(w', 1\right) = 2i + 1 < 2d + 1 = t,$$

a contradiction. We denote the properties $(*)$ and $(**)$ as "tree structure up to $\mathrm{niv}(d)$" (see also Figure 2.3).

We prove by induction: $|\mathrm{niv}(i)| = k(k-1)^{i-1}, 1 \leq i \leq d$. For $i = 1$, the claim is clear. Let $i > 1$. For the number of edges connecting nodes from $\mathrm{niv}(i-1)$ to nodes from $\mathrm{niv}(i)$:

$$|\mathrm{niv}(i-1)|(k-1) = |\mathrm{niv}(i)|.$$

By the induction hypothesis,

$$|niv(i)| = k(k-1)^{i-2}(k-1) = k(k-1)^{i-1}.$$

So, Γ has at least $\sum_{i=0}^{d} |niv(i)| = 1 + \sum_{i=1}^{d} k(k-1)^{i-1}$ nodes.

ii) Now let $t = 2d+2$ be even. We choose a neighbor z of node 1 and define for $i = 0, \ldots, d$ the disjoint sets

$$\mathrm{niv}(i) := \mathrm{niv}_{\Gamma,\{1,z\}}(i) := \{v \in \underline{n} \mid \mathrm{dist}(v, \{1, z\}) = i\}.$$

For $1 \leq i \leq d$, each node $w \in \mathrm{niv}(i)$ has exactly one neighbor in $\mathrm{niv}(i-1)$. If w were adjacent to two nodes $v_1, v_2 \in \mathrm{niv}(i-1)$, then Γ would have a cycle of length less than or equal to

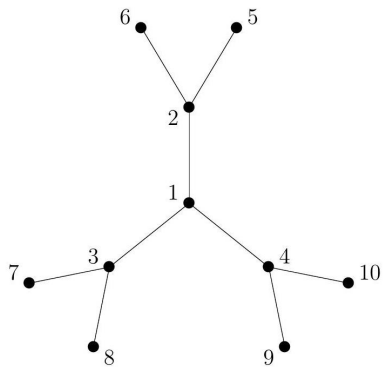
$$\mathrm{dist}(1, z) + \mathrm{dist}(v_1, \{1, z\}) + \mathrm{dist}(v_1, w) + \mathrm{dist}(w, v_2) + \mathrm{dist}(v_2, \{1, z\}) = 2i + 1 \leq 2d + 1 < t$$

a contradiction.

Furthermore, for $1 \leq i < d$, each node $w \in \mathrm{niv}(i)$ has exactly $k-1$ neighbors in $\mathrm{niv}(i+1)$. If w were adjacent to a node $w' \in \mathrm{niv}(i)$, then Γ would have a cycle of length less than or equal to

$$\mathrm{dist}(1, z) + \mathrm{dist}(w, \{1, z\}) + \mathrm{dist}\left(w, w'\right) + \mathrm{dist}\left(w', \{1, z\}\right) = 2i + 2 < 2d + 2 = t,$$

a contradiction. We have tree structure up to $\mathrm{niv}(d)$ (see also Figure 2.4).



figTree0

FIGURE 2.3. The tree $T_{3,5}$.

Claim: $|\operatorname{niv}(i)| = 2(k-1)^i, 0 \le i \le d$. For $i = 0$, the claim is clear. Let $i > 0$. For the number of edges connecting nodes from $\operatorname{niv}(i-1)$ to nodes in $\operatorname{niv}(i)$, we have

$$|\operatorname{niv}(i-1)|(k-1) = |\operatorname{niv}(i)|$$

By the induction hypothesis,

$$|\operatorname{niv}(i)| = 2(k-1)^{i-1}(k-1) = 2(k-1)^i.$$

So Γ has at least

$$\sum_{i=0}^{d} |\operatorname{niv}(i)| = \sum_{i=0}^{d} 2(k-1)^i$$

nodes.

□

In the proof, we consider trees $T_{k,t}$, as shown in Figures 2.3 and 2.4. We want to precisely define $T_{k,t}$:

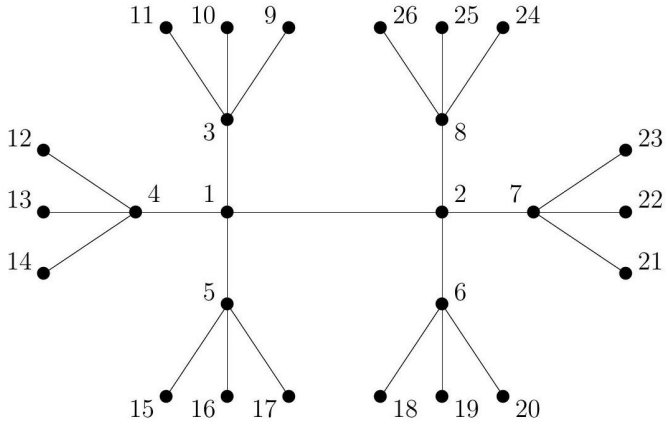
{defSpanningTree}

DEFINITION 2.2.5. Let $k \ge 2, t \ge 3$. For odd $t = 2d + 1$, let $T_{k,t} \in \mathcal{G}_{f_0(k,t)}$ denote the tree with root 1 and the following properties:

a) Each leaf v has distance $\operatorname{dist}(v, 1) = d$ to node 1.
b) Each internal node w has $\deg(w) = k$.
c) $T_{k,t} \in \mathcal{G}_{f_0(k,t)}$ is canonical as per Definition 1.4.6.

For even $t = 2d + 2$, let $T_{k,t} \in \mathcal{G}_{f_0(k,t)}$ denote the tree with root 1 and the following properties:

a) Nodes 1 and 2 are adjacent, and for each leaf v, $\operatorname{dist}(v, \{1, 2\}) = d$.
b) Each internal node w has $\deg(w) = k$.



figTree

FIGURE 2.4. The tree $T_{4,6}$.

c) $T_{k,t} \in \mathcal{G}_{f_0(k,t)}$ is canonical as per Definition 1.4.6.

More generally, let $\bar{T}_{n,k,t}$ denote the numbered graph with $n \geq f_0(k,t)$ nodes obtained from $T_{k,t}$ by adding $n - f_0(k,t)$ degree 0 nodes.

{rkLevels}

REMARK 2.2.6. Let $n = f_0(k,t)$. Consider the disjoint sets:

$$\mathrm{niv}(i) := \begin{cases} \mathrm{niv}_{T_{k,t},1}(i) = \left\{ v \in \underline{n} \mid \mathrm{dist}_{T_{k,t}}(v,1) = i \right\}, & \text{if } t = 2d+1 \text{ is odd}, \\ \mathrm{niv}_{T_{k,t},\{1,2\}}(i) = \left\{ v \in \underline{n} \mid \mathrm{dist}_{T_{k,t}}(v,\{1,2\}) = i \right\}, & \text{if } t = 2d+2 \text{ is even}, \end{cases}$$

where $0 \leq i \leq d$. The following statements about $T_{k,t}$ are easy to see:

i) $v \in \mathrm{niv}(i), w \in \mathrm{niv}(j), 0 \leq i < j \leq d \Longrightarrow v < w$;
ii) $v, v' \in \mathrm{niv}(i), w, w' \in \mathrm{niv}(i+1), 0 \leq i < d$, w a child of v, w' a child of v', $v < v' \Longrightarrow w < w'$
iii) $v, v' \in \mathrm{niv}(i), w, w' \in \mathrm{niv}(j), 0 \leq i < j \leq d$, w a successor of v, w' a successor of $v', v < v' \Longrightarrow w < w'$
iv) The nodes with degree k are $1, \ldots, f_0(k, t-2)$, where $f_0(k,2) := 2$ and $f_0(k,1) := 1$.
v) The leaves are $f_0(k,t-2)+1, \ldots, f_0(k,t)$.
vi) For odd $t > 3$, $f_0(k,t-2)+1, \ldots, f_0(k,t-1)$ are the leaves that are successors of node 2. $f_0(k,t-1)+1$ is the smallest leaf that is a successor of node 3.
vii) For even $t > 4$, $f_0(k,t-1)+1, \ldots, f_0(k,t)$ are the leaves that are successors of node 2. $f_0(k,t-2)+1$ is the smallest leaf that is a successor of node 3.

{lmSpanningTree}

LEMMA 2.2.7. *Let* $\Gamma \in \mathrm{rep}\left(S_n \backslash\backslash \mathcal{R}_{n,k,t}\right)$. *Then* $\bar{T}_{n,k,t} \subset \Gamma$. *Furthermore, for every* $w \in \underline{n}$, *there exists a* $\pi \in S_n$ *such that* $w^\pi = 1$ *and* $\bar{T}_{n,k,t} \subset \Gamma^\pi$.

PROOF. Let d be as in Remark 2.2.6. As in the proof of Theorem 2.2.4, we show that Γ possesses tree structure up to $\mathrm{niv}(d)$. $\Gamma|_{f_0(k,t-2)} := \{(x,y) \in \Gamma \mid x \leq f_0(k,t-2)\} \in \mathcal{G}_{f_0(n,k)}$ is thus a tree that satisfies conditions a) and b) from Definition 2.2.5.

Since Γ is canonical and $\Gamma|_{f_0(k,t-2)} \subset \Gamma, \Gamma|_{f_0(k,t-2)} < \Gamma$, it follows that $\Gamma|_{f_0(k,t-2)}$ is canonical (this can be shown as in Theorem 3.1.2). Thus, c) is also satisfied, i.e., $\bar{T}_{n,k,t} = \Gamma|_{f_0(k,t-2)}$ and the first part of the claim is proven.

•2.2.2: rg: *label* •2.2.2

Let $w \in \underline{n}$ be given, $t = 2d+1$ odd. We consider the disjoint sets $\mathrm{niv}_{\Gamma,w}(i) = \left\{v \in \underline{n} \mid \mathrm{dist}_\Gamma(v,w) = i\right\}, 0 \leq i \leq d$. As in the proof of Theorem 2.2.4, we show that tree structure exists up to $\mathrm{niv}_{\Gamma,w}(d)$. Let the neighbors of w be $w_1, \ldots, w_k$, and the nodes in $\mathrm{niv}_{\Gamma,w}(2)$ be $w_{1,1}, \ldots, w_{1,k-1}, w_{2,1}, \ldots\ldots, w_{k,k-1}$.

Generally, we denote the $k-1$ neighbors in $\mathrm{niv}_{\Gamma,w}(i+1)$ of node $w_{j_1,\ldots,j_i} \in \mathrm{niv}_{\Gamma,w}(i)$ as $w_{j_1,\ldots,j_i,1}, \ldots, w_{j_1,\ldots,j_i,k-1}$.

We define $\pi \in S_n$ such that ${w^\pi} = 1, {w_1}^\pi = 2, \ldots, {w_k}^\pi = k+1\,(= f_0(k,3))$ and for $w_{j_1,\ldots,j_i} \in \mathrm{niv}_{\Gamma,w}(i)$, ${w_{j_1,\ldots,j_i}}^\pi = f_0(k,2i-1) + 1 + \sum_{l=1}^{i}(k-1)^{i-l}(j_l - 1)$. Then $\bar{T}_{n,k,t} = \Gamma^\pi|_{f_0(k,t-2)}$, and the claim follows.

The case where $t = 2d+2$ is even can be treated similarly: We choose a neighbor z of w and consider

$$\mathrm{niv}_{\Gamma,\{w,z\}}(i) = \left\{v \in \underline{n} \mid \mathrm{dist}_\Gamma(v,\{w,z\}) = i\right\}.$$

Γ has tree structure up to $\mathrm{niv}(d)$. Let the neighbors of w be $w_1, \ldots, w_{k-1}$, and the neighbors of z be $w_k, \ldots, w_{2k-2}$. Generally, we denote the $k-1$ neighbors in $\mathrm{niv}_{\Gamma,\{w,z\}}(i+1)$ of node $w_{j_1,\ldots,j_i} \in \mathrm{niv}_{\Gamma,\{w,z\}}(i)$ as $w_{j_1,\ldots,j_i,1}, \ldots, w_{j_1,\ldots,j_i,k-1}$. We define $\pi \in S_n$ such that ${w^\pi} = 1, {z^\pi} = 2, {w_1}^\pi = 3, \ldots, {w_{2k-2}}^\pi = 2k\,(= f_0(k,4))$ and for $w_{j_1,\ldots,j_i} \in \mathrm{niv}_{\Gamma,w}(i)$, ${w_{j_1,\ldots,j_i}}^\pi = f_0(k,2i)+1+\sum_{l=1}^{i}(k-1)^{i-l}(j_l-1)$. Then $\bar{T}_{n,k,t} = \Gamma^\pi|_{f_0(k,t-2)}$, and the claim follows. □

REMARK 2.2.8. As can be seen from Table 1, the function values $f_0(k,t)$ grow very rapidly even for small k and t (even exponentially depending on t). Therefore, function values for f are only known in a few cases. Table 2 contains some of these function values, along with the numbers $\left|S_{f(k,t)} \backslash\backslash \mathcal{R}_{f(k,t),k,t}\right|$ of (k,t)-cages in parentheses. The numbers are based on the information in [Won] and [BriMcK], as well as the program created as part of this thesis. In some other cases, only $f(k,t)$ is known, but not the number of (k,t)-cages. Recently, Brendan McKay and Wendy Myrvold calculated that $f(3,11) = 112$.

$f_0(k,t)$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
$t = 3$	4	5	6	7	8
$t = 4$	6	8	10	12	14
$t = 5$	10	17	26	37	50
$t = 6$	14	26	42	62	86
$t = 7$	22	53	106	187	302
$t = 8$	30	80	170	312	518
$t = 9$	46	161	426	938	1824
$t = 10$	62	242	682	1563	3120

TABLE 1. Function values for f_0.

{tabF0}

$f(k,t)$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
$t = 5$	10(1)	19(1)	30(4)	40(1)	50(1)
$t = 6$	14(1)	26(1)	42(1)	62(1)	90(1)
$t = 7$	24(1)	$f(k,t)$ or number of (k,t)-cages not known			
$t = 8$	30(1)				
$t = 9$	58(18)				
$t = 10$	70(3)				

TABLE 2. Function values for f and number of (k,t)-cages.

{tabCages}

CHAPTER 3

Order-Preserving Generation

{ch3}

We now want to focus specifically on the construction tasks formulated in Remark 1.3.8. We employ the principle of order-preserving generation, first described by Read in [Re]. The basis for this is the total order explained in Definition 1.4.1 on the set of numbered graphs.

An algorithm is developed to construct, in lexicographically increasing order, all numbered k-regular graphs that are eligible for the canonical representative system $\mathrm{rep}_<\left(S_n \backslash\backslash \mathcal{R}_{n,k}\right)$. Additionally, properties such as "connected" and "girth $\geq t$" can be taken into account.

A canonicality test must decide in which cases the generated graphs are canonical orbit representatives. Since such a test is associated with significant effort (see Chapter 4), it is crucial to limit the set of test candidates. To achieve this, necessary criteria for the canonicity of a k-regular numbered graph are presented, which help avoid many non-canonical candidates during the process of edge insertion.

3.1. Read's Method

First, let's formulate Read's method in general: Let $(Z, \leq)$ be a totally ordered set and G a group that acts on Z. Then,

$$\mathrm{rep}_<(G \backslash\backslash Z) = \left\{z \in Z \mid \forall g \in G : z \leq z^g\right\}$$

is a canonical transversal of the orbits of G on Z. The order-preserving generation according to Read is described by the following theorem:

{thmRead}

THEOREM 3.1.1. *Let $Z = \bigcup_{i=1}^{n} Z_i$, ${Z_i}^G \subseteq Z_i$, $Z_i \cap Z_j = \emptyset$ $(i \neq j)$, and P be an algorithm that, for every $z \in \mathrm{rep}_<(G \backslash\backslash Z_i)$, generates a set $P(z) \subseteq Z$, such that*

- $\forall i,\ \mathrm{rep}_<(G \backslash\backslash Z_{i+1}) \subseteq \bigcup_{z \in \mathrm{rep}_<(G \backslash\backslash Z_i)} P(z)$,
- $\forall i,\ \forall z \in \mathrm{rep}_<(G \backslash\backslash Z_{i+1}),\ \exists! z' \in \mathrm{rep}_<(G \backslash\backslash Z_i) : z \in P\left(z'\right)$.

Then, the desired transversal is obtained by:

i) Generate $\mathrm{rep}_<(G \backslash\backslash Z_1)$ and set $\mathrm{rep}_<(G \backslash\backslash Z) := \mathrm{rep}_<(G \backslash\backslash Z_1)$.
ii) For $i = 1, \ldots, n$, iterate through $rep_<(G \backslash\backslash Z_i)$ with z, and generate $P(z)$ during this process. Continue iterating through $P(z)$

with w and check if w is minimal in its orbit. If yes, set $\mathrm{rep}_<(G\backslash\backslash Z) := \mathrm{rep}_<(G\backslash\backslash Z) \cup w$.

We can apply this theorem to the set $\mathcal{G}_n$. To do this, we use the lexicographic order on $\mathcal{G}_n$ from Definition 1.4.1. The following theorem can also be found in [HKLMW].

{thmSubgraph}

THEOREM 3.1.2. *If* $\Gamma \in \mathrm{rep}_<(S_n\backslash\backslash\mathcal{G}_n)$ *and* $\Gamma_1 \subset \Gamma$ *with* $\Gamma_1 < \Gamma$, *then* $\Gamma_1 \in \mathrm{rep}_<(S_n\backslash\backslash\mathcal{G}_n)$.

PROOF. Let $\Gamma = \Gamma_1 \cup \Gamma_2$ and $\Gamma_1 \notin \mathrm{rep}_<(S_n\backslash\backslash\mathcal{G}_n)$ with $\Gamma_1 < \Gamma$. Then, $\Gamma_1^\pi < \Gamma_1$ for some $\pi \in S_n$. If $\Gamma_1^\pi = \{e_1, \ldots, e_t\}$ with $e_1 < e_2 < \ldots < e_t$, then there exists an i such that $\Gamma_1 = \left\{e_1, \ldots, e_{i-1}, e_i', \ldots, e_t'\right\}, e_{i-1} < e_i' < \ldots < e_t'$ and $e_i < e_i'$. It holds $\Gamma^\pi = \Gamma_1^\pi \cup \Gamma_2^\pi \supseteq \{e_1, \ldots, e_i\}$, so that $\Gamma^\pi < \Gamma_1 < \Gamma$, a contradiction to $\Gamma \in \mathrm{rep}_<(S_n\backslash\backslash\mathcal{G}_n)$. □

REMARK 3.1.3. To apply Theorem 3.1.1, we need to define the set $P(\Gamma)$ for a representative as

$$P(\Gamma) := \left\{\Gamma \cup \{e\} \mid e > \max\left\{e' \in \Gamma\right\}\right\}.$$

Although there is no complexity analysis for Theorem 3.1.1, this approach is quite efficient. In the following, we will see how to use this method for generating regular graphs.

3.2. Application to Regular Graphs

{defPRead}

DEFINITION 3.2.1. For $\Gamma \in \mathcal{G}_n$, let $P^{(1)}(\Gamma) := P(\Gamma)$ and

$$P^{(i+1)}(\Gamma) := \bigcup_{\Gamma' \in P^{(i)}(\Gamma)} P\left(\Gamma'\right), i \geq 1.$$

If $\max\left\{e' \in \Gamma'\right\} = (n-1, n)$, then let $P\left(\Gamma'\right) := \varnothing$. We denote by $Q(\Gamma)$ the set

$$Q(\Gamma) := \bigcup_{i \geq 1} P^{(i)}(\Gamma)$$

Clearly, for this set, $Q(\Gamma) = \left\{\Gamma' \in \mathcal{G}_n \mid \Gamma \subset \Gamma' \wedge \Gamma < \Gamma'\right\}$. The following statement is almost trivial:

{lmInitialData}

LEMMA 3.2.2. *We have* $\mathrm{rep}_<\left(S_n\backslash\backslash\mathcal{R}_{n,k}\right) \subseteq Q(\{(1,2)\})$.

PROOF. Let $\Gamma \in \mathrm{rep}_<\left(S_n\backslash\backslash\mathcal{R}_{n,k}\right), \Gamma = \{e_1, \ldots, e_m\}, e_1 < \ldots < e_m, m = \frac{nk}{2}$. Then, $e_1 = (1,2)$. For $i = 2, \ldots, m$, let $\Gamma^{(i)} := \{e_1, \ldots, e_i\} \subseteq \Gamma$.

We show by induction that $\Gamma^{(i)} \in P^{(i-1)}(\{(1,2)\}) : \Gamma^{(2)} = \{e_1, e_2\} \in P(\{e_1\}) = P^{(1)}\{e_1\}$, $\Gamma^{(i+1)} = \left(\Gamma^{(i)} \cup \{e_{i+1}\}\right) \in P\left(\Gamma^{(i)}\right) \subseteq P^{(i)}(\Gamma)$, since by assumption $\Gamma^{(i)} \in P^{(i-1)}$. □

Based on Theorem 3.1.2, we can describe the following function, which, after calling ordrek($\{(1,2)\}$), outputs the set $\mathrm{rep}_<\left(S_n \backslash\backslash \mathcal{R}_{n,k}\right)$.

{algoOrdrek}

Algorithm 1 Construction of all canonical regular graphs containing Γ

Objective: To construct all canonical regular graphs that contain Γ as a subgraph, where each graph has n nodes of degree k.
Input: Graph Γ
procedure ORDREK(Γ)
1: Verify if, by adding further edges, Γ can be extended to a regular graph with n nodes of degree k. If not, **return**.
2: Check if Γ is part of the representation $\mathrm{rep}_<\left(S_n \backslash\backslash \mathcal{G}_n\right)$. If not, **return**.
3: If Γ belongs to $\mathcal{R}_{n,k}$, **output** Γ and **return**.
4: Iterate through $P(\Gamma)$ in ascending order with Γ' and execute ORDREK(Γ') to construct the canonical regular graphs incrementally.
end procedure

The algorithm actually delivers all elements of $\mathrm{rep}_<\left(S_n \backslash\backslash \mathcal{R}_{n,k}\right)$: Let Γ be a sought-after representative, $\Gamma^{(i)} \subseteq \Gamma$ as in the proof of Lemma 3.2.2. Then, each $\Gamma^{(i)}$ passes the test under 1. According to Theorem 3.1.2, for each i: $\Gamma^{(i)} \in \mathrm{rep}_<\left(S_n \backslash\backslash \mathcal{G}_n\right)$.

It is easy to see that the algorithm traverses the graphs in lexicographically increasing order. Thus, it cannot happen that a representative is output multiple times.

To decide under 2. whether $\Gamma \in \mathrm{rep}_<\left(S_n \backslash\backslash \mathcal{G}_n\right)$, a canonicality test (see Chapter 4) must be performed. Under 1., it must be clear that $\Gamma \in \mathcal{G}_{n,k}$. Furthermore, additional criteria are checked here:

{lmCrit1}

LEMMA 3.2.3. *Let* $(x,y) = \max\{e \in \Gamma\}, y > n-k$, *and* $\deg(x) < k$. *If* $n - y < k - \deg(x)$, *then* $Q(\Gamma) \cap \mathcal{R}_{n,k} = \emptyset$.

PROOF. $k - \deg(x)$ neighbors are still needed for x. Only the nodes $y+1, \ldots, n$ are eligible as neighbors. Thus, there are still $n-y$ possible neighbors. Therefore, it must hold that $n - y \geq k - \deg(x)$. □

{lmCrit2}

LEMMA 3.2.4. *Let* $(x,y) = \max\{e \in \Gamma\}, x \geq n-k$, *and* $\deg(x) = k$. *If for some* i *with* $y < i \leq n$ *it holds that* $n - x - 1 < k - \deg(i)$, *then* $Q(\Gamma) \cap \mathcal{R}_{n,k} = \emptyset$.

PROOF. If $\deg(i) = k$, then the condition is never satisfied (the left side of the inequality is always ≥ 0). So, let $\deg(i) < k$. Then node i still needs $k - \deg(i)$ more neighbors. Nodes $x+1, \ldots, i-1, i+1, \ldots, n$

are available for this purpose, so there are $n-x-1$ possible neighbors. Therefore, it must hold that $n-x-1 \geq k-\deg(i)$. □

REMARK 3.2.5. Even stronger conditions like those in Lemma 3.2.3 and 3.2.4 can be formulated. However, in practice, the effort required to check these conditions increases more than the savings that can be achieved. Furthermore, such criteria have the common disadvantage that they become more effective only when inserting the last edges and thus play an increasingly less important role for larger n. Nevertheless, based on this, one can design a fairly efficient algorithm for small n.

We will examine the following pseudocode of the function ORDREK(x, y, v) more closely in Algorithm 3.2. The following data structures are required:

- A data structure to store the graph. One option is a two-dimensional array of size $n \times n$ for the adjacency matrix, or an array of length nk, where for each node i, its neighbors are listed starting from position $(i-1)k+1$, called a neighborhood list. In the present implementation, multiple data structures are maintained simultaneously to ensure optimal access in different situations. The functions INSERT(x, y) and DELETE(x, y) then handle the insertion and deletion of edge (x, y) in accordance with the chosen data structure.
- A vector *deg* of length n, which contains the degrees of the nodes. While these could be computed for each query based on the adjacency matrix or neighborhood list, time is saved by updating the relevant entries in *deg* only when adding or deleting an edge using INSERT(x, y) or DELETE(x, y).
- Variables x, y, and v, which are passed during function call. Here, (x, y) is the last inserted edge, and v is the smallest node with degree 0 before the insertion of edge (x, y).
- Global variables n and k, and the local variable i.

Before the first call, the data structure for the graph is initialized to contain $\bar{T}_{n,k,3}$ from Definition 2.2.5 (using Lemma 2.2.7). Similarly, the node degrees are adjusted: $\deg[1] = k, \deg[2] = \ldots = \deg[k+1] = 1, \deg[i] = 0$ for $i = k+2, \ldots, n$. Upon calling ORDREK$(1, k+1, k+1)$, the representative system $\mathrm{rep}_{<}\left(S_n \backslash\backslash \mathcal{R}^*_{n,k}\right)$ of connected k-regular graphs with n nodes is output.

In lines 2-4 and 5-11, conditions for Lemma 3.2.3 or 3.2.4 are checked. If true, a recursion step can be taken back. In lines 12-14, x is assigned the smallest node with degree less than k. If no such node exists, then $x = n$. Lines 15-17 ensure that v contains the smallest node

{algoOrdrek2}

Algorithm 2 Detailed ORDREK Algorithm

```
1: procedure ORDREK(x, y, v)
2:     if y > n − k ∧ deg[x] < k ∧ n − y < k − deg[x] then
3:         return
4:     end if
5:     if x ≤ n − k ∧ deg[x] = k then
6:         for i = y + 1 to n do
7:             if n − x − 1 < k − deg[i] then
8:                 return
9:             end if
10:        end for
11:    end if
12:    while x < n ∧ deg[x] = k do
13:        x := x + 1
14:    end while
15:    if v ≤ y then
16:        v := y + 1
17:    end if
18:    if x = v then
19:        return
20:    end if
21:    if KATEST() = 0 then
22:        return
23:    end if
24:    if x = n ∧ deg[x] = k then
25:        OUTPUT()
26:    end if
27:    y := x
28:    while y < n do
29:        y := y + 1
30:        if deg[y] < k then
31:            INSERT(x, y)
32:            ORDREK(x, y, v)
33:            DELETE(x, y)
34:        end if
35:    end while
36:    return
37: end procedure
```

with degree 0 (if there is no node with this property, then $v = n+1$ afterwards). If this node matches x, then no further edge can be inserted, ensuring the graph remains connected as per Definition 1.3.3 (Lemma 3.2.6 is used here). If only connected k-regular graphs are to be constructed, no final connectivity test needs to be performed. By omitting lines 18-20, regular graphs with multiple connected components can also be generated. Lines 21-23 test for canonization. The function KATEST() is explained in 4.2.5. Lines 24–26 determine whether a k-regular graph with n nodes has been found. If so, it is output.

•3.2.1: rg: label

Lines 27-35 control the insertion of a new edge. All eligible edges are traversed in ascending order (using Lemma 3.2.7). Line 30 ensures that y does not yet have degree k. Lemma 3.2.8 could also be applied here. Line 31 inserts the new edge (x, y). Line 32 recursively calls ORDREK(x, y, v), and once that is completed, (x, y) is deleted again.

We do not want to miss the opportunity to add the lemmas used:

{lmCrit3}

LEMMA 3.2.6. *Let* $x = \min\{i \in \underline{n} \mid \deg_\Gamma(i) < k\}$, $\deg(x) = 0$.
$\Longrightarrow P(\Gamma) \cap \mathcal{G}^*_{n,k} = \varnothing, Q(\Gamma) \cap \mathcal{G}^*_{n,k} = \varnothing, Q(\Gamma) \cap \mathcal{R}^*_{n,k} = \varnothing$.

PROOF. Clear. □

{lmCrit4}

LEMMA 3.2.7. *Let* $x = \min\{i \in \underline{n} \mid \deg_\Gamma(i) < k\}$. *Then for* $x' = x+1, \ldots, n-1$, $y = x'+1, \ldots, n$: $Q\left(\Gamma \cup \left\{(x', y)\right\}\right) \cap \mathcal{R}_{n,k} = \varnothing$.

PROOF. For all $\Gamma' \in Q\left(\Gamma \cup \left\{(x', y)\right\}\right)$: $\deg_\Gamma(x) < k$. □

{lmCrit5}

LEMMA 3.2.8. *If* Γ *has nodes of degree 0, let* $v = \min\{i \in \underline{n} \mid \deg_\Gamma(i) = 0\}$. *Also, let* $x = \min\{i \in \underline{n} \mid \deg_\Gamma(i) < k\}$ *and* $x \neq v$. *Then for* $v < y \leq n$: $(\Gamma \cup \{(x, y)\}) \notin \mathrm{rep}_<(S_n \backslash\backslash \mathcal{G}_n)$.

PROOF. $(\Gamma \cup \{(x, y)\})^{(v,y)} = (\Gamma \cup \{(x, v)\}) < (\Gamma \cup \{(x, y)\})$. This is a special case of Theorem 3.3.2. □

REMARK 3.2.9. Algorithm 3.2 performs a canonization test nearly after each addition of a new edge. Such a test is very expensive since, in principle, all elements π of S_n must be traversed to verify if $\Gamma \leq \Gamma^\pi$ (see Chapter 4). This implies unnecessary time expenditure in two ways: Firstly, candidates that ultimately pass must be tested multiple times. Secondly, it may happen that although graphs pass the canonization test multiple times, it is later found that they cannot be completed to regular graphs. In both cases, the effort to test canonization was in vain. It has proven to be much more effective to perform the canonization test only on graphs from $\mathcal{R}_{n,k}$. Otherwise, only necessary canonization conditions are checked. This is important to avoid

producing too many candidates for the final canonization test. Such conditions will be developed in the next sections.

3.3. The Row Criterion

DEFINITION 3.3.1. Let $\Gamma \in \mathcal{G}_n, \Gamma_i := \{(i,v) \in \Gamma\}, 1 \le i < n$. Thus, $\Gamma = \bigcup_{i=1}^{n-1} \Gamma_i, \Gamma_i \cap \Gamma_j = \varnothing \ (i \ne j)$ and with $1 \le i < j < n, e \in \Gamma_i, e' \in \Gamma_j \Rightarrow e < e'$.

We define $C_1 := C_{S_n}(\{1\})$ and for $1 \le i < n$:

$$N_i := N_{C_i}(\Gamma_i) = \left\{\pi \in C_i \mid \Gamma_i^\pi = \Gamma_i\right\},$$
$$C_{i+1} := C_{N_i}(\{1,\dots,i+1\}) = \left\{\pi \in N_i \mid (i+1)^\pi = i+1\right\}$$

{thmSemicanonical}

THEOREM 3.3.2. *Let* $\Gamma \in \mathrm{rep}_<(S_n \backslash\backslash \mathcal{G}_n)$. *Then*

$$\forall i < n : \forall \pi \in C_i : \quad \Gamma_i \le {\Gamma_i}^\pi \quad (*).$$

PROOF. Indirectly: Let i_0 be the smallest i for which the assertion is not fulfilled, i.e., $\exists \tau \in C_{i_0} : {\Gamma_{i_0}}^\tau < \Gamma_{i_0}$. Because $C_{i_0} \le N_j \forall j < i_0$, we have ${\Gamma_j}^\tau = \Gamma_j \forall j < i_0$. Thus, $\Gamma^\tau < \Gamma$, which contradicts the canonization of Γ. □

DEFINITION 3.3.3. If $\Gamma \in \mathcal{G}_n$ has the property (*) from Theorem 3.3.2, then we call Γ semicanonical.

REMARK 3.3.4. The term "semicanonical" is also defined in the same way in [Gru2]. Our goal is to ensure that only semicanonical candidates are constructed when inserting the edges. This requires a row-wise approach: The edges from Γ_i correspond to the ones to the right of the diagonal in the i-th row a_i of the adjacency matrix A of Γ. So, the edges in Γ_i must be set in such a way that $\Gamma_i \le {\Gamma_i}^\pi \forall \pi \in C_i$. This is equivalent to $a_i \succeq a_i^\pi \forall \pi \in C_i$, where a_i^π is the i-th row of A^π. We will work out what this means for the row a_i.

A major advantage of this method is that the required normalizer N_i can be easily determined based on a_i and C_i. Both the C_i and the N_i are Young subgroups of S_n. The C_i can be determined particularly easily:

{rkCentralizer}

REMARK 3.3.5. The centralizer $C_{S_n}(\{1,\dots,i\})$ of the digits $1,\dots,n$ can also be viewed as a Young subgroup of S_n: Let $\mu = (\mu_1,\dots,\mu_{i+1}) \vDash n$ with $\mu_1 = \dots = \mu_i = 1$ and $\mu_{i+1} = n - i$. Then $S_\mu = C_{S_n}(\{1,\dots,i\})$.

If $S_\lambda \le C_{S_n}(\{1,\dots,i\})$ holds for $\lambda = (\lambda_1,\dots,\lambda_m) \vDash n$, then λ has the form

$$\lambda = (\underbrace{1,\dots,1}_{i}, \lambda_{i+1}, \dots \lambda_m)$$

Then $C_{S_\lambda}(\{1,\dots,i+1\})$ can also be written as a Young subgroup: For

$$\zeta := \begin{cases} (\underbrace{1,\dots,1}_{i+1}, \lambda_{i+1}-1, \lambda_{i+2}, \dots, \lambda_m) & \text{if } \lambda_{i+1} > 1 \\ \lambda & \text{otherwise} \end{cases}$$

we have $S_\zeta = C_{S_\lambda}(\{1,\dots,i+1\})$.

The N_i are also Young subgroups. To determine these, some combinatorial considerations are needed:

{defPartition}

DEFINITION 3.3.6. Let $\mu = (\mu_1,\dots,\mu_s) \vDash n$ be a partition of n. We consider the operation of S_μ on the set of $0/1$-vectors of length n :

$$\{0,1\}^{\underline{n}} \times S_\mu \longrightarrow \{0,1\}^{\underline{n}}, \quad (x_1,\dots,x_n)^\pi := \left(x_{\pi(1)},\dots,x_{\pi(n)}\right)$$

Let $x \in \{0,1\}^{\underline{n}}$. We call x canonical with respect to μ if

$$\forall \pi \in S_\mu : \quad x \geq x^\pi$$

A permutation $\sigma \in S_\mu$ is called a canonizing permutation of x with respect to μ if x^σ is canonical with respect to μ.

{lmCanonical}

LEMMA 3.3.7. *With the notations from Definition 3.3.6, it holds:*

x is canonical w.r.t. $\mu \iff \forall i = 1,\dots,s\ \forall j,j' \in \underline{n}_i^\mu$ with $j < j' : x_j \geq x_{j'}$.

PROOF. $\pi \in S_\mu$ only permutes entries of x within the blocks given by μ. Thus, both implications are clear. □

REMARK 3.3.8. If $x \in \{0,1\}^{\underline{n}}$ is canonical with respect to μ, then within the blocks given by μ, the ones of x are on the left and the zeros are on the right. On the other hand, we can determine a canonizing permutation of x by sorting the ones to the left and the zeros to the right within the μ-blocks. We can easily define a merging mapping in this way.

Our interest shall now focus on the stabilizer of canonical $x \in \{0,1\}^{\underline{n}}$, i.e., the set $\left\{\pi \in S_\mu \mid x^\pi = x\right\}$. These are precisely the permutations that exchange the entries within the μ-blocks such that the ones remain on the left and the zeros on the right. This is described by the following definition.

efCanonicalRefinment}

DEFINITION 3.3.9. Let $\mu = (\mu_1,\dots,\mu_s) \vDash n$ be a partition and $x \in \{0,1\}^{\underline{n}}$ canonical with respect to μ. Furthermore, for $i = 1,\dots,s$, let $\mu_i^1 := \left|\left\{j \in \underline{n}_i^\mu : x_j = 1\right\}\right|$ be the number of ones of x in the i-th block with respect to μ, and $\mu_i^0 := \left|\left\{j \in \underline{n}_i^\mu : x_j = 0\right\}\right|$ be the number of zeros of x in the i-th block with respect to μ. From the partition

$$\left(\mu_1^1, \mu_1^0, \mu_2^1, \mu_2^0, \dots, \mu_s^1, \mu_s^0\right) \vDash n$$

we remove all components μ_i^1 and μ_i^0 that have the value 0. We call the resulting partition $\eta \vDash n$ the canonical refinement of μ with respect to x.

{lmSeta}

LEMMA 3.3.10. *With the designations from Definition 3.3.9, we have:*

$$S_\eta = \left\{\pi \in S_\mu \mid x^\pi = x\right\}$$

PROOF. Clear by construction of η. □

Before applying these tools to the rows of the adjacency matrix A, let us give a fictitious example:

EXAMPLE 3.3.11. Let $n = 10, \mu = (3,1,2,4)$, and $x = (0,1,0,1,0,0,1,0,1,0)$. For better clarity, we mark the blocks of x given by μ with vertical lines:

$$x = (0,1,0,|1,|0,0,|\ 1,0,1,0).$$

A canonizing permutation is $\pi = (1,2)(8,9) \in S_\mu$, where

$$x^\pi = (1,0,0,|1,|0,0,|\ 1,1,0,0).$$

From the counts of ones and zeros in the μ-blocks of x, we obtain

$$\left(\mu_1^1, \mu_1^0, \mu_2^1, \mu_2^0, \mu_3^1, \mu_3^0, \mu_4^1, \mu_4^0\right) = (1,2,1,0,0,2,2,2)$$

and by omitting the components with value 0, we get the canonical refinement η of μ with respect to x:

$$\eta = (1,2,1,2,2,2)$$

{rdCanonizing}

REMARK AND DEFINITION 3.3.12. We can view the row $a_i = (a_1, \ldots, a_n)$ of A as a 0/1 vector. For a permutation $\pi \in S_n$, ${a_i}^\pi$ is the i-th row of A^π:

$${a_i}^\pi := \left(a_{\pi(i),\pi(1)}, \ldots, a_{\pi(i),\pi(n)}\right).$$

Let $\zeta = (\zeta_1, \ldots, \zeta_i, \zeta_{i+1}, \ldots) \vDash n$ be a partition with $\zeta_j = 1, j = 1, \ldots, i$. Then $S_\zeta \leq C_{S_n}(\{1, \ldots, i\})$. For $\tau \in S_\zeta$,

$$a_i^\tau = \left(a_{\tau(i),\tau(1)}, \ldots, a_{\tau(i),\tau(n)}\right) = \left(a_{i,1}, \ldots, a_{i,i}, a_{i,\tau(i+1)}, \ldots, a_{i,\tau(n)}\right)$$

i.e., τ only permutes the entries of row a_i from position $i+1$ within the ζ-blocks. We call row a_i canonical with respect to ζ if

$$\forall \tau \in S_\zeta : \quad a_i \geq {a_i}^\tau.$$

A permutation $\sigma \in S_\zeta$ is called a canonizing permutation of row a_i with respect to ζ if ${a_i}^\sigma$ is canonical with respect to ζ. This concept will play a central role in the canonicity test later.

We now come to the

Application 3.3.13. For our algorithm, we apply the insights gained as follows: Let $\Gamma_i = \varnothing$ and $C_i = S_{\zeta^{(i)}}$ be known (initially, according to Remark 3.3.5, we set $\zeta^{(1)} = (1, n-1)$). The insertion of edges into Γ_i should be done so that a_i is canonical (as in 3.3.12) with respect to $\zeta^{(i)}$. This is achieved by placing the ones in such a way that within the $\zeta^{(i)}$-blocks, they always appear to the left of the zeros. Then, according to Lemma 3.3.7, we have

$$\forall \pi \in S_{\zeta^{(i)}} : \quad a_i \geq {a_i}^{\pi}$$

This is equivalent to

$$\forall \pi \in C_i : \quad \Gamma_i \leq {\Gamma_i}^{\pi}$$

Next, we compute the canonical refinement $\eta^{(i)}$ of $\zeta^{(i)}$ with respect to a_i. For the induced Young subgroup, according to Lemma 3.3.10:

$$S_{\eta^{(i)}} = \left\{\pi \in S_{\zeta^{(i)}} \mid {a_i}^{\pi} = a_i\right\} = \left\{\pi \in C_i \mid \Gamma_i^{\pi} = \Gamma_i\right\} = N_i.$$

Finally, we determine $C_{i+1} = C_{S_{\eta^{(i)}}}(\{1, \ldots, i+1\})$ as in 3.3.5. We obtain $\zeta^{(i+1)}$ with

$$S_{\zeta^{(i+1)}} = C_{i+1}$$

and we can proceed with Γ_{i+1}.

Example 3.3.14. For the case $n = 8$ and $k = 3$, there are 10 ways to fill the first three rows of the adjacency matrix such that they

satisfy the row criterion. The $\zeta^{(i)}$ blocks are indicated by vertical lines:

$$\begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$

$$\begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

$$\begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

$$\begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$

$$\begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} \qquad \begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

3.4. The Tail Criterion

Now, we turn to a condition that requires regularity in particular. The method presented in this section was developed by Gunnar Brinkmann. Lemma 3.4.3 can also be found in a similar form in [?].

{thmGirth}

THEOREM 3.4.1. *Let* $\Gamma \in \mathrm{rep}_<(S_n \setminus \mathcal{R}_{n,k})$. *Then node 1 lies on a tail cycle of* Γ. *Nodes 2 and 3 belong to the same tail cycle.*

PROOF. Follows from Lemma 3.4.3. □

REMARK 3.4.2. If we drop the condition that Γ is regular, Theorem 3.4.1 becomes false. Figure 3.1 shows an example from $\mathrm{rep}_<(S_n \setminus \mathcal{G}_n)$, where node 1 does not lie on a tail cycle.

{lmGirth}

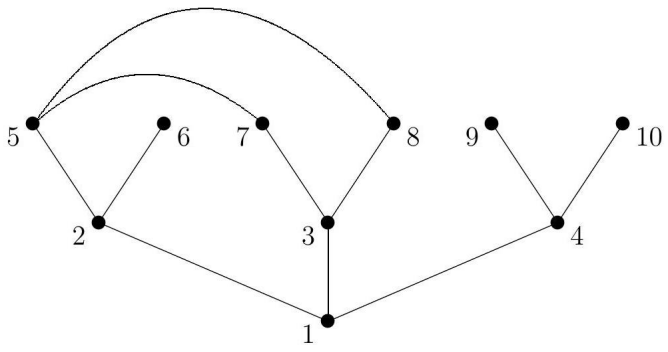
LEMMA 3.4.3. *If* $\Gamma \in \mathrm{rep}_<(S_n \setminus \mathcal{R}_{n,k})$, *then the function girthcheck*(Γ) *describes a tail cycle in* Γ, *on which node 3 also lies.*

•*3.4.1*

•*3.4.1:* **rg:** *A terminer*

PROOF. Let $t = \mathrm{girth}(\Gamma)$.

i) $t = 2d + 1$ odd. That node $u = f_0(k, t-2) + 1$ is reached is clear from Remark 2.2.6 and Lemma 2.2.7. u is the smallest successor of node 2 in $N_{\Gamma,1}(d) = \{v \in \underline{n} \mid \mathrm{dist}_\Gamma(v, 1) = d\}$, or $u = 2$ if $t = 3$.



figTail

FIGURE 3.1. An example illustrating the tail criterion.

Algorithm 3 ... {algoGirth}

Objective: ...
Input: Graph Γ
procedure GIRTHCHECK(Γ)
1: Start at node 1 and go to node 2.
2: Upon reaching node i, move to the smallest neighbor, excluding the predecessor on the current path.
3: Repeat step 2 until reaching node 1.
end procedure

We consider the neighbors of u. u has a neighbor in $N_{\Gamma,1}(d-1)$, the predecessor on the previously described path. The next larger neighbor is to be found in $N_{\Gamma,1}(d)$. Due to the tail width, u cannot be adjacent to a successor of node 2 in $N_{\Gamma,1}(d)$. The smallest successor $w = f_0(k, t-1)+1$ of node 3 in $N_{\Gamma,1}(d)$ can be a neighbor of u.

Claim: $(u, w) \in \Gamma$. Suppose $(u, w) \notin \Gamma$. We choose a tail cycle $v_0, v_1, \ldots, v_d, v_{d+1}, \ldots, v_{t-1}, v_0$. Lemma 2.2.7 yields a $\pi \in S_n$ such that:

$$v_0^\pi = 1, v_d^\pi = f_0(k, t-2)+1, v_{d+1}^\pi = f_0(k, t-1)+1.$$

Then, $(f_0(k, t-2)+1, f_0(k, t-1)+1) \in \Gamma^\pi < \Gamma$, as Γ^π and Γ coincide in the smaller edges (these are precisely the edges of $\bar{T}_{n,k,t}$). This contradicts the canonicity of Γ, and the claim is proven. Upon reaching node w, one returns to 1 via 3.

ii) $t = 2d+2$ even. According to Remark 2.2.6 and Lemma 2.2.7, node $w = f_0(k, t-1)+1$ is reached. w is the smallest successor of node 2 in $N_{\Gamma,\{1,2\}}(d) = \{v \in \underline{n} \mid \mathrm{dist}_\Gamma(v, \{1,2\}) = d\}$. The smallest neighbor of w is in $N_{\Gamma,\{1,2\}}(d-1)$. The next larger neighbor of w comes from

$N_{\Gamma,\{1,2\}}(d)$. This is $u = f_0(k, t-2)+1$. As shown in (i), $(u, w) \in \Gamma$. The rest is clear, as u is a successor of node 3, or $u = 3$ if $t = 4$.

□ {corGirth}

COROLLARY 3.4.4. *If* $\Gamma \in \mathrm{rep}_<(S_n \setminus \mathcal{R}_{n,k})$, $\Gamma' \subset \Gamma$ *with* $\Gamma' < \Gamma$, *and* Γ' *contains exactly one cycle, then this cycle is described by* `girthcheck` (Γ'), *and it coincides with the cycle constructed by* `girthcheck` (Γ).

PROOF. Let $\Gamma = \{e_1, \ldots, e_m\}, e_1 < \ldots < e_m$, $t = \mathrm{girth}(\Gamma)$, $\Gamma' = \{e_1, \ldots, e_{m'}\}$, $m' < m$. It holds that $\bar{T}_{k,t} = \{e_1, \ldots, e_s\}$, where $s = f_0(k, t) - 1$. According to the proof of Lemma 3.4.3, $e_{s+1} = (u, w)$. Since Γ' contains a cycle, $m' \geq s + 1$, and the claim follows. □

APPLICATION 3.4.5. We can use the newly gained insights to reduce the set of candidates:

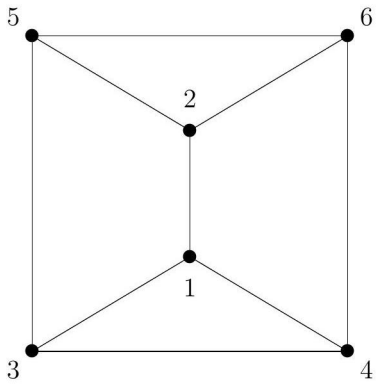
If Γ does not yet have a cycle and by adding edge e, a first cycle is closed, we check whether `girthcheck` $(\Gamma \cup \{e\})$ describes this cycle and whether node 3 lies on it. If both conditions are met, we note the length of this cycle and continue adding more edges. Otherwise, $\Gamma \cup \{e\}$ can be discarded.

If Γ already has a cycle and `girth` $(\Gamma) > 3$ was noted, when adding edge e, it must be checked whether `girth` $(\Gamma \cup \{e\}) < \mathrm{girth}(\Gamma)$. If this is the case, $\Gamma \cup \{e\}$ can be discarded. Figure 3.1 shows a situation that occurs in our algorithm from Section 3.2: The newly added edge is $(5, 8)$. The girth was previously 5 and is now 4. In the recursion, a step can be taken back.

The fact that a new cycle has been closed can be easily determined. In the algorithm from Section 3.2, this is the case when ORDREK(x, y, v) is called with $y < v$. If `girthcheck` (Γ) is successful for the first cycle, then the length of this cycle is equal to the number of traversed nodes (where node 1 is counted only once, of course). In most cases, cycles already exist. Then, starting from the two nodes of the newly added edge (x, y), a "breadth-first" search is conducted to determine if a new cycle with length $< \mathrm{girth}(\Gamma)$ has been formed. This is only relevant if `girth` $(\Gamma) > 3$.

EXAMPLE 3.4.6. Figure 3.2 shows an element $\Gamma \in \mathcal{R}_{6,3}$ where the tail criterion can be applied. Γ is semi-canonical, would be constructed according to our previous method, and would now need to be tested for canonicity. However, it is already evident after inserting edge $(3, 4)$, according to Corollary 3.4.4, that no representative can be constructed.

Not only does one save a canonicity test, but also the insertion of additional edges (see also Figure 3.6).



figAvoidable

FIGURE 3.2. An avoidable candidate.

			without Tail Criterion			with Tail Criterion		
n	k	Graphs	Candidates	Factor	Time	Candidates	Factor	Time
10	3	19	250	13.16	0.0 s	61	3.21	0.0 s
12	3	85	2999	35.28	0.6 s	513	6.04	0.2 s
14	3	509	41856	82.23	8.2 s	5825	11.44	1.9 s
16	3	4060	665765	163.98	127 s	75873	18.69	22.5 s
18	3	41301	11887987	287.83	2258 s	1147792	27.79	319 s
9	4	16	268	16.75	0.0 s	159	9.94	0.0 s
10	4	59	2224	37.69	0.4 s	1066	18.07	0.2 s
11	4	265	20045	75.64	3.3 s	7940	29.96	1.7 s
12	4	1544	194437	125.93	31.2 s	64745	41.93	12.7 s
13	4	10778	2028053	188.17	319 s	577369	53.57	111 s

{tabTail}

TABLE 1. Effect of the Tail Criterion.

REMARK 3.4.7. The tail criterion can lead to a considerable reduction in the candidate set, as shown in Table 3.4. In this table:

- Graphs: the number of connected k-regular graphs with n nodes;
- Candidates: the number of numbered connected k-regular graphs that need to be tested for canonicity;
- Factor: the ratio Candidates/Graphs;
- Time: the CPU time required on a PC 486 DX2/66.

It would be interesting to investigate whether stronger statements than Theorem 3.4.1 are valid and how they would affect the candidate set.

REMARK 3.4.8. We can now also construct $\mathrm{rep}_<\left(S_n \backslash\backslash \mathcal{R}_{n,k,t}\right)$ with $t > 3$. To do this, we need to initialize the data structure for the graph in such a way that it contains $\bar{T}_{n,k,t}$ at the beginning. If edge

e closes the first cycle such that girthcheck $(\Gamma \cup \{e\})$ runs correctly, then this cycle has length $\geq t$. Thus, all subsequently closed cycles also have length $\geq t$, and the regular graph at the end has the desired girth. However, it should be noted that the current implementation regarding the construction of regular graphs with large girth can still be significantly improved.

3.5. The Learning Effect

Using the results of a failed canonicity test, we can further reduce the candidate set or, in the words of [Gru1]:

Man kann etwas lernen ! (We can learn something!)

The following theorem describes how, in the case of a non-canonical test candidate, we obtain a necessary criterion for the canonicity of its lexicographical successors.

{thmGrund}

THEOREM 3.5.1. *Let* $\Gamma \in \mathcal{G}_n$ *be non-canonical,* $\Gamma = \{e_1, \ldots, e_t\}$ *with* $e_1 < e_2 < \ldots < e_t$.

$\Longrightarrow \exists \pi \in S_n : \Gamma^\pi < \Gamma$

$\Longrightarrow \exists i < t : \Gamma^\pi = \left\{e_1, \ldots, e_i, e'_{i+1}, \ldots, e'_t\right\}$ *with* $e'_{i+1} < e_{i+1}$.

It holds that $\left\{e_1^{\pi^{-1}}, \ldots, e_i^{\pi^{-1}}, e_{i+1}^{\pi^{-1}}\right\} = \left\{e_{j_1}, \ldots, e_{j_{i+1}}\right\}$ *with* $1 \leq j_l \leq t \forall l = 1, \ldots, i+1$.

Let $r := \max\{j_1, \ldots, j_{i+1}, i+1\} \Longrightarrow$ *all* $\tilde{\Gamma} \in \mathcal{G}_n$ *with* $\tilde{\Gamma} = \{e_1, \ldots, e_r, \tilde{e}_{r+1}, \ldots, \tilde{e}_s\}$, $e_1 < \ldots < e_r < \tilde{e}_{r+1} < \ldots < \tilde{e}_s$ *are also non-canonical.*

PROOF. Since $r \geq i+1$, we have $\tilde{\Gamma} = \{e_1, \ldots, e_i, e_{i+1}, \ldots, e_r, \tilde{e}_{r+1}, \ldots, \tilde{e}_s\}$. Furthermore,

$\tilde{\Gamma}^\pi = \{e_1^\pi, \ldots, e_r^\pi, \tilde{e}_{r+1}^\pi, \ldots, \tilde{e}_s^\pi\} \supseteq \left\{e_{j_1}^\pi, \ldots, e_{j_{i+1}}^\pi\right\} = \left\{e_1, \ldots, e_i, e'_{i+1}\right\} \Longrightarrow$ $\tilde{\Gamma}^\pi < \tilde{\Gamma}$ because $e'_{i+1} < e_{i+1}$. □

APPLICATION 3.5.2. For practical application, this means: if a non-canonical candidate has been tested, determine r as in the above theorem and go back in the recursion ORDREK(x, y, v) until $e_r \notin \Gamma$. The smaller e_r is, the more candidates can be neglected.

{rkCanonicity}

REMARK 3.5.3. However, a negatively performed canonicity test does not always yield the permutation that provides the greatest learning effect. This is because the exhaustive canonicity test terminates at the first found $\pi \in S_n$ with the property $\Gamma^\pi < \Gamma$ (see Chapter 4). Overall, it would not be profitable to always search for the element that achieves the maximum learning effect.

We want to specify how we calculate e_r (the notations from Theorem 3.5.1 are retained): In the case of a negatively performed canonicity test, we obtain π with $\Gamma^\pi < \Gamma$ and the edge $(v,w) = {e_{i+1}}'$. Then $\left\{e_1^{\pi^{-1}}, \ldots, e_i^{\pi^{-1}}, e_{i+1}^{\pi^{-1}}\right\} = \left\{(x,y)^{\pi^{-1}} \mid (x,y) \in \Gamma, (x,y) < (v,w)\right\} \cup \left\{(v,w)^{\pi^{-1}}\right\}$ and

$$e_r = \max\left(\left\{(x,y)^{\pi^{-1}} \mid (x,y) \in \Gamma, (x,y) < (v,w)\right\} \cup \left\{(v,w)^{\pi^{-1}}\right\} \cup \{(v,w)\}\right).$$

Thus, determining the edge e_r requires very little effort. Given this, it is surprising what effect can be achieved with this learning effect. For demonstration, consider the following example.

{exLearning}

EXAMPLE 3.5.4. Let us consider $\Gamma \in \mathcal{R}_{9,4}$ given by

$$\Gamma = \{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,5),\\ (4,5),(4,8),(5,9),(6,7),(6,8),(6,9),(7,8),(7,9),(8,9)\}$$

Γ is non-canonical. With $\pi = (3,2)$, we have

$$\Gamma^\pi = \{(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),\ldots\} < \Gamma$$

Using the notations from Remark 3.5.3, $(v,w) = (2,4)$. We obtain

$$e_r = \max(\{(1,2),(1,3),(1,4),(1,5),(2,3)\} \cup \{(3,4)\} \cup \{(2,4)\}) = (3,4)$$

In this case, 37 candidates can be skipped:

{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,5),(4,6),(4,7),(5,8),(5,9),(6,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,5),(4,6),(4,8),(5,7),(5,9),(6,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,5),(4,6),(4,8),(5,8),(5,9),(6,7),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,5),(4,8),(4,9),(5,6),(5,7),(6,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,5),(4,8),(4,9),(5,6),(5,8),(6,7),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,5),(4,8),(4,9),(5,8),(5,9),(6,7),(6,8),(6,9),(7,8),(7,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,5),(4,7),(5,8),(5,9),(6,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,5),(4,8),(5,7),(5,9),(6,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,5),(4,8),(5,8),(5,9),(6,7),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,6),(4,8),(5,7),(5,8),(5,9),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,7),(4,8),(5,6),(5,8),(5,9),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,7),(4,8),(5,7),(5,8),(5,9),(6,8),(6,9),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,8),(4,9),(5,6),(5,7),(5,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,8),(4,9),(5,6),(5,8),(5,9),(6,7),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,8),(4,9),(5,7),(5,8),(5,9),(6,7),(6,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,6),(4,8),(4,9),(5,7),(5,8),(5,9),(6,8),(6,9),(7,8),(7,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,5),(4,6),(5,7),(5,9),(6,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,5),(4,6),(5,8),(5,9),(6,7),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,5),(4,8),(5,6),(5,9),(6,7),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,5),(4,9),(5,6),(5,7),(6,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,5),(4,9),(5,6),(5,8),(6,7),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,5),(4,9),(5,6),(5,9),(6,7),(6,8),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,5),(4,9),(5,8),(5,9),(6,7),(6,8),(6,9),(7,8),(7,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,7),(5,6),(5,8),(5,9),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,8),(5,6),(5,7),(5,9),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,8),(5,7),(5,8),(5,9),(6,7),(6,9),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,9),(5,6),(5,7),(5,8),(6,9),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,9),(5,6),(5,7),(5,9),(6,8),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,9),(5,6),(5,8),(5,9),(6,7),(7,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,9),(5,7),(5,8),(5,9),(6,7),(6,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,9),(5,7),(5,8),(5,9),(6,7),(6,9),(7,8),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,6),(4,9),(5,7),(5,8),(5,9),(6,8),(6,9),(7,8),(7,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,8),(4,9),(5,6),(5,7),(5,8),(6,7),(6,9),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,8),(4,9),(5,6),(5,7),(5,9),(6,7),(6,8),(7,9),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,8),(4,9),(5,6),(5,7),(5,9),(6,7),(6,9),(7,8),(8,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,8),(4,9),(5,6),(5,7),(5,9),(6,8),(6,9),(7,8),(7,9)}
{(1,2),(1,3),(1,4),(1,5),(2,3),(2,6),(2,7),(3,4),(3,8),(4,8),(4,9),(5,6),(5,8),(5,9),(6,7),(6,9),(7,8),(7,9)}

3.5.0.1. *Remark:* Example 3.5.4 is by no means a carefully chosen special case. As can be seen from Table 3.5, a significant reduction in the candidate set is achieved in all cases.

Particularly noteworthy is the fact that when using the learning effect, the ratio of candidates to representatives decreases as n increases (except for some exceptions for "small" n), which is also confirmed by Table 5.3. There is a conjecture that with increasing n, the number of candidates asymptotically approaches the number of sought-after representatives.

•3.5.1 •3.5.1: rg: label

3.6. Generation Trees

We can visualize the generation algorithm as a tree: the root is $\bar{T}_{n,k,t}$ and every graph formed by inserting a new edge is another node. In the deepest level, there are only elements from $\mathcal{R}^*_{n,k,t}$. On the following

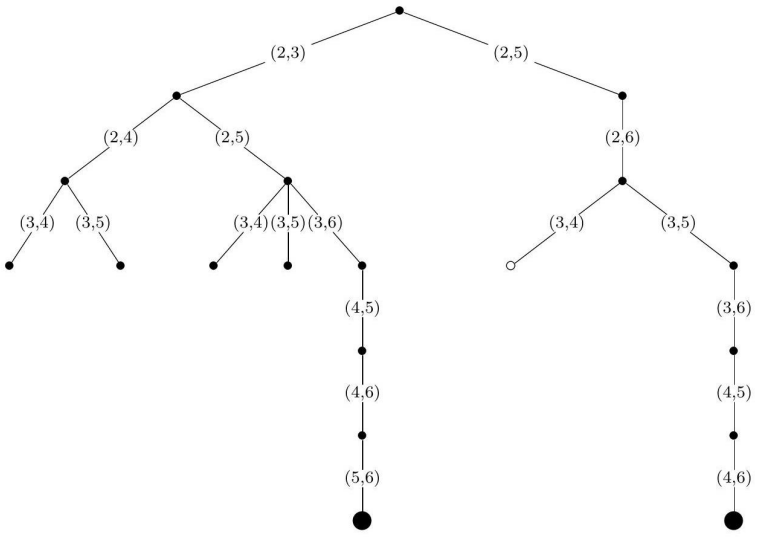
			without learning effect			with learning effect		
n	k	Graphs	Candidates	Factor	Time	Candidates	Factor	Time
10	3	19	61	3.21	0.0 s	37	1.95	0.0 s
12	3	85	513	6.04	0.2 s	214	2.52	0.1 s
14	3	509	5825	11.44	1.9 s	1406	2.76	1.0 s
16	3	4060	75873	18.69	22.5 s	10432	2.57	8.9 s
18	3	41301	1147792	27.79	319 s	96279	2.33	95.7 s
9	4	16	159	9.94	0.0 s	57	3.56	0.0 s
10	4	59	1066	18.07	0.2 s	219	3.71	0.1 s
11	4	265	7940	29.96	1.7 s	997	3.76	0.6 s
12	4	1544	64745	41.93	12.7 s	5194	3.36	3.2 s
13	4	10778	577369	53.57	111 s	33139	3.07	22.5 s

{tabLearning}

TABLE 2. Effect of the Learning Effect.

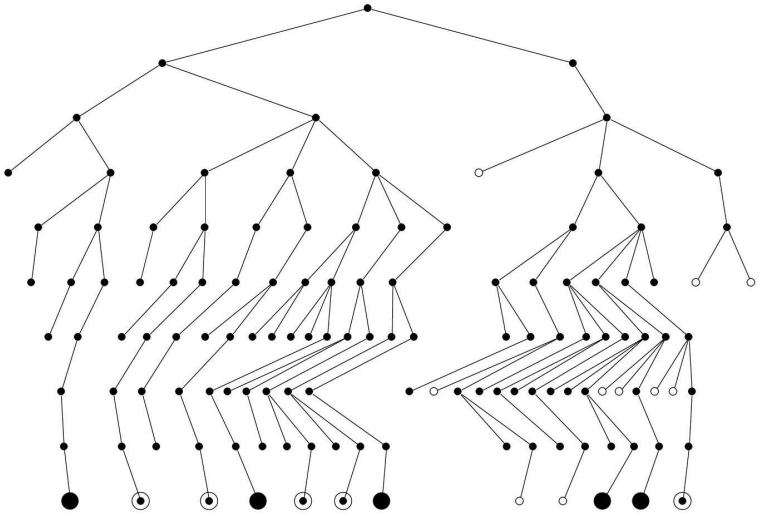
pages, several such generation trees for different n, k, and t are shown. The interested reader can understand the successive insertion of edges and the application of different criteria based on these illustrations. In the manageable examples, it is noted which edge is inserted. The labeling of the nodes has the following meaning:

- ∘: Tail criterion not fulfilled.
- •: Canonicity test performed with negative result.
- ●: Canonicity test performed with positive result.



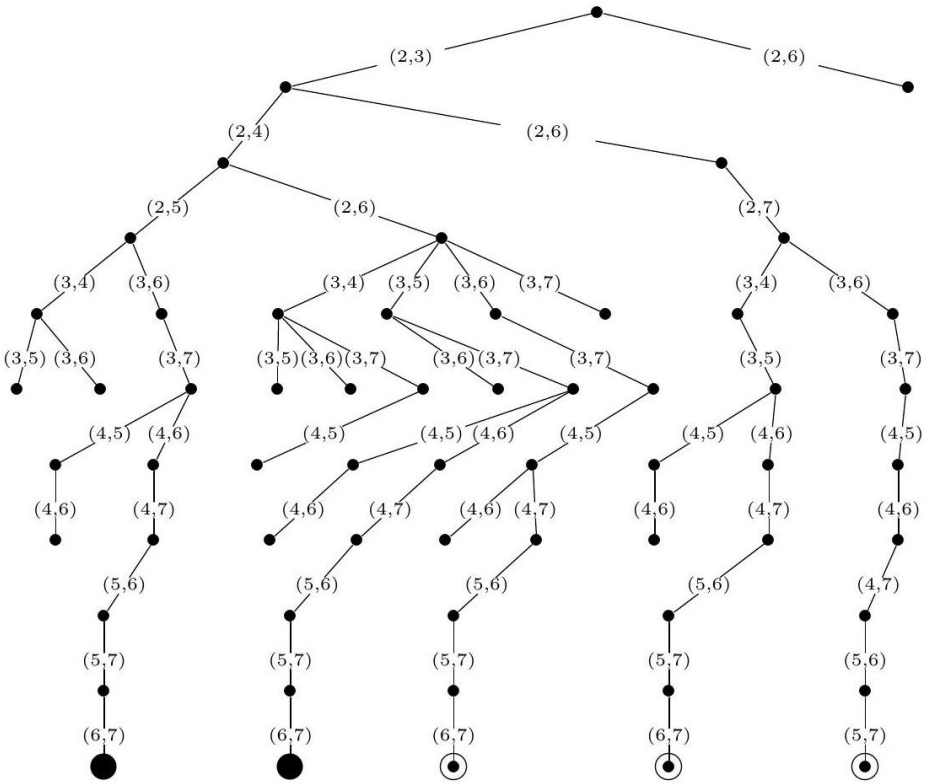
{figGeneration63}

FIGURE 3.3. The generation tree for $n = 6, k = 3$.



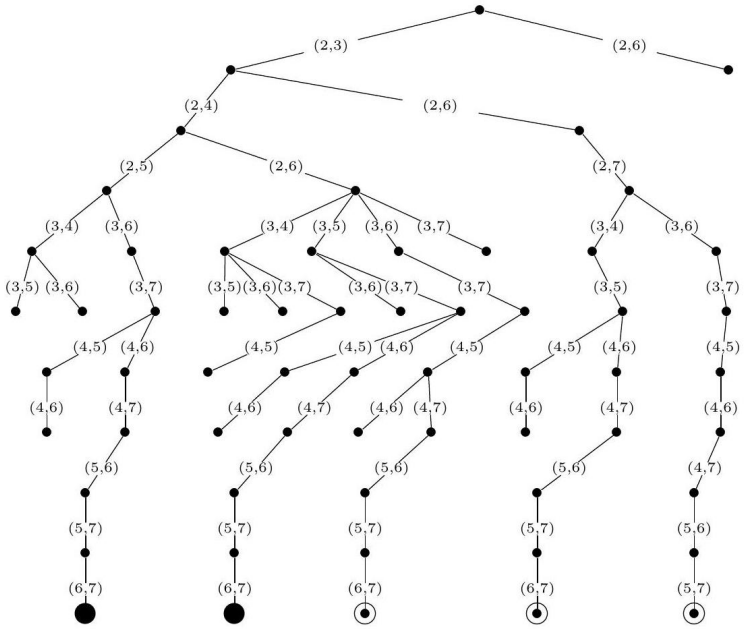
{figGeneration83}

FIGURE 3.4. The generation tree for $n = 8, k = 3$.



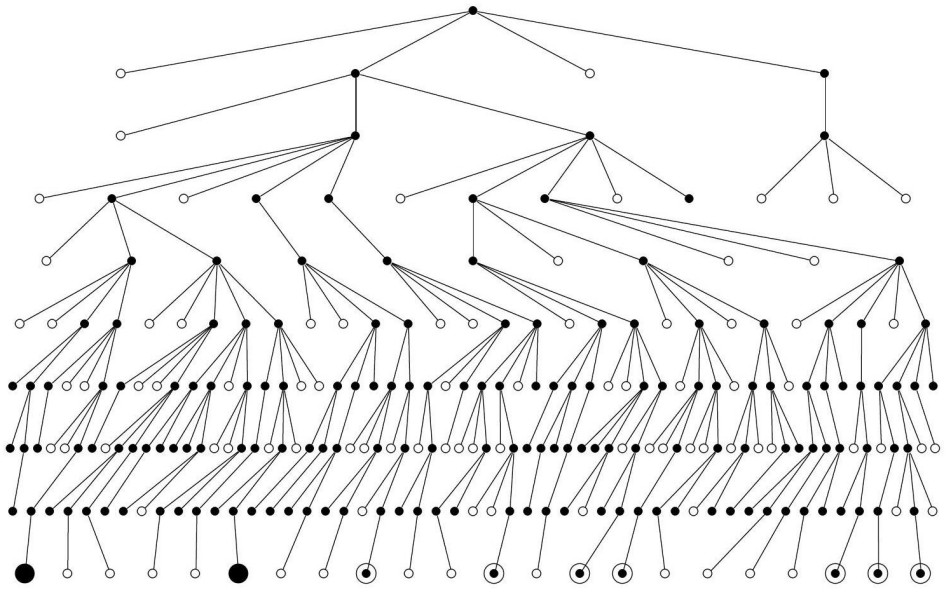
{figGeneration74}

FIGURE 3.5. The generation tree for $n = 7, k = 4$.



{figGeneration1035}

FIGURE 3.6. The generation tree for $n = 10, k = 3, t = 5$.



{figGeneration1235}

FIGURE 3.7. The generation tree for $n = 12, k = 3, t = 5$.

CHAPTER 4

Canonicality Test

{ch4}

This test is the most challenging part of the entire algorithm. In the end, the efficiency of the implementation will largely depend on whether it is possible to limit the effort for this test. It must be decided whether for a given candidate $\Gamma \in \mathcal{R}_{n,k}$ holds

$$\Gamma \leq \Gamma^{\pi} \quad \forall \pi \in S_n.$$

First, a traversal strategy for permutation groups is presented, which is based on the use of a series of special subgroups, called Sims chains. Furthermore, it is described how a system of generators of the automorphism group can be found, leading to an initial reduction of the test. Finally, we will make use of the fact that only semi-canonical candidates need to be tested. We use the normalizers defined in Section 3.3 and the concept of canonizing permutation to further refine the canonicality test in such a way that only a few permutations need to be considered from the outset. During the row-wise comparison of the adjacency matrices of Γ and Γ^{π}, it can then be decided after each row whether parts of the test can be neglected. This procedure is demonstrated with two detailed examples.

4.1. Sims Chains

To answer the question posed above, we first need to develop some techniques for dealing with permutation groups. In particular, we need:

- A strategy to traverse all elements of S_n.
- A data structure to manage the automorphism group of $\Gamma \in \mathrm{rep}_{<}\left(S_n \backslash\backslash \mathcal{R}_{n,k}\right)$ as a subgroup of S_n in the memory of a computer.

Both tasks can be solved using the principle of Sims chains (see [**?**]).

DEFINITION 4.1.1. Let $U \leq S_{\underline{n}}$ and for $i = 1, \ldots, n-1$

$$U_i := C_U(\{1, \ldots, i\})$$

be the centralizer of the digits 1,.., i. The subgroup chain

$$U =: U_0 \geq U_1 \geq U_2 \geq \ldots \geq U_{n-1} = \{id\}$$

{thmCosets}

is called the Sims chain of U.

THEOREM 4.1.2. *Let* $1 \le i < n, l(i) := |U_{i-1}/U_i|$ *and* $\{u_{i,1}, \ldots, u_{i,l(i)}\}$ *be a representative system of* U_{i-1}/U_i*, where* $u_{i,1} = id$*. Then there exists the following disjoint decomposition of* U_{i-1} *into left cosets of* U_i *in* U_{i-1}*:*

$$U_{i-1} = \bigcup_{j=1}^{l(i)} u_{i,j} U_i$$

PROOF. The statement follows from Theorem 1.2.6 and Example 1.2.7. □

COROLLARY 4.1.3. *Every group element* $u \in U$ *is uniquely representable as*

$$u = \prod_{i=1}^{n-1} u_{i,j_i}, \quad 1 \le j_i \le l(i), 1 \le i < n$$

PROOF. We prove the assertion for $u \in U_i, n-1 \ge i \ge 0$ by induction on i.

For $i = n-1$, $u = id$ and the assertion is clear.

Let $u \in U_{i-1}$. According to Theorem 4.1.2, $\exists! j_i \le l(i) : u \in u_{i,j_i} U_i$. Then $\exists! u' \in U_i : u = u_{i,j_i} u'$. By the induction hypothesis, u' is uniquely representable as $u' = u_{i+1,j_{i+1}} \ldots u_{n-1,j_{n-1}}$. Thus, $u = u_{i,j_i} u_{i+1,j_{i+1}} \ldots u_{n-1,j_{n-1}} = \prod_{\nu=1}^{n-1} u_{\nu,j_\nu}$, where $j_\nu = 1$ for $\nu < i$. □

EXAMPLE 4.1.4. For $U = S_n$, we can explicitly specify the decomposition of the centralizers into left cosets:

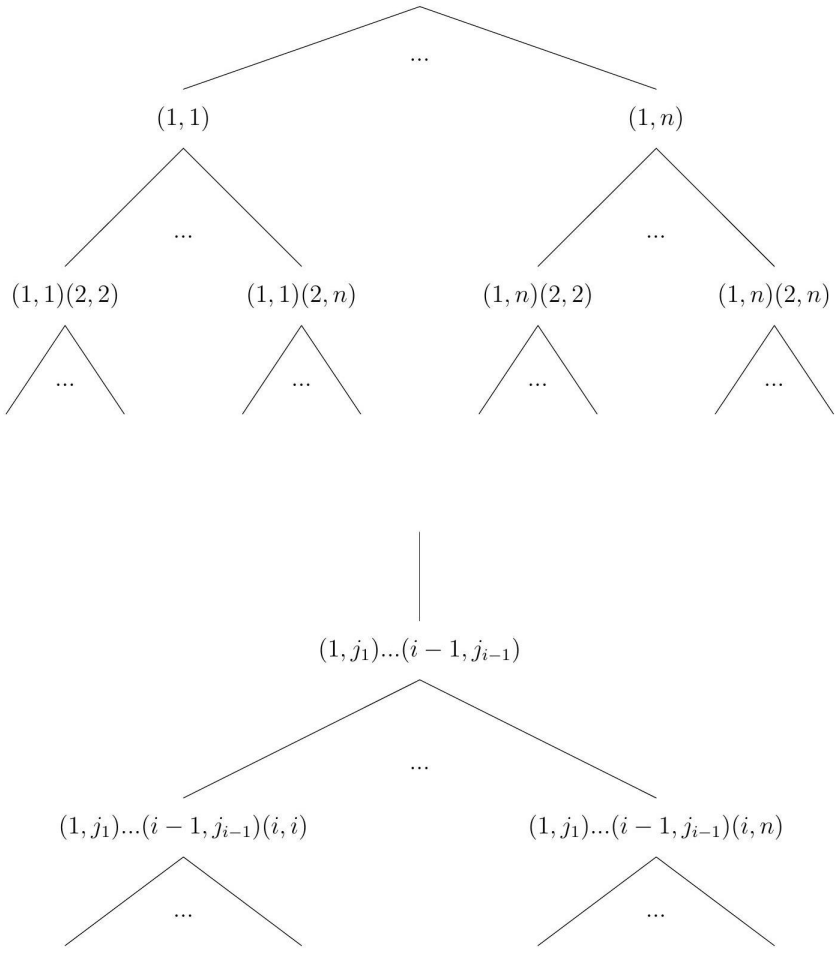
$$U_{i-1} = \bigcup_{j=i}^{n} (i,j) U_i, \quad i = 1, \ldots, n-1$$

The coset representatives are transpositions. Every permutation $\pi \in S_n$ is uniquely representable in the form:

$$\pi = \prod_{i=1}^{n-1} (i, j_i), \quad i \le j_i \le n, 1 \le i < n.$$

(i,i) denotes the identity.

REMARK 4.1.5. Following Example 4.1.4, we can imagine S_n as a tree of depth $n-1$ (see Figure 4.1). The leaves of the group tree are the elements of S_n. The nodes at depth 1 correspond to the coset representatives of U_0/U_1 and the leaves of the branch from node $(1,j)$ are the elements of $(1,j)U_1$. In general, for nodes π at depth i, the leaves of the branch from π are the elements of πU_i. At the left edge,



{figGroupTree}

FIGURE 4.1. The group tree of S_n.

we have in each depth id. Thus, for $j = 1, \ldots, n$, the first j leaves from the left are precisely the elements of U_{n-j}.

{appNaive}

APPLICATION 4.1.6. By traversing the group tree in a "depth-first" manner, we reach every leaf (=group element) successively (from left to right). This procedure provides a simple canonization test given in Algorithm 4.1.6.

Obviously, this algorithm tests all $n!$ permutations and is therefore very inefficient. The following considerations aim to reduce the number of permutations that need to be tested.

Algorithm 4 Naive Test Algorithm {algoNaive}

1: **procedure** NAIVETEST(Γ)
2: Traverse S_n using the depth-first method and check if:
3:
$$\Gamma \leq \Gamma^{(1,j_1)(2,j_2)\ldots(n-1,j_{n-1})}, \quad i \leq j_i \leq n, \quad 1 \leq i < n$$
4: **if** the condition is not met **then**
5: **return** (not canonical)
6: **end if**
7: **end procedure**

4.2. Calculation of the Automorphism Group

For many theoretical and practical applications, it is important not only to have a representative system of $S_n \backslash\backslash \mathcal{R}_{n,k}$ but also to know the automorphism groups of the computed structures (see [**?**, **?**] or Section 5.2). We will now see how we can find a generating system of the automorphism group $\mathrm{Aut}(\Gamma)$ for a canonical $\Gamma \in \mathcal{R}_{n,k}$. To do this, let's take a closer look at the traversal strategy from 4.1.6. We recognize:

•4.2.1 •*4.2.1: rg: label*

- It starts with the smallest centralizer $U_{n-1} = \{id\}$ and then moves on to the next larger one:
$$U_{n-1} \longrightarrow U_{n-2} \longrightarrow \ldots \longrightarrow U_0$$
- This is done by traversing the cosets U_{i-1}/U_i after U_i:
$$U_i \longrightarrow (i, i+1)U_i \longrightarrow \ldots \longrightarrow (i, n)U_i.$$

After that, all elements of U_{i-1} have been visited. When finding the first automorphism in each coset, we can apply the following theorem:

THEOREM 4.2.1. *Let $j > i$ and $\pi \in U_i$ be the first permutation found when traversing $(i, j)U_i$ with $\Gamma^{(i,j)\pi} = \Gamma$. Then $\Gamma \leq \Gamma^{(i,j)\sigma} \forall \sigma \in U_i$.*

PROOF. It holds that $\Gamma^{\tau} \geq \Gamma \ \forall \tau \in U_i$ (by definition of naivetest). Let $\sigma \in U_i$, then $\Gamma^{(i,j)\sigma} = \Gamma^{\pi^{-1}\sigma} \geq \Gamma$. □

REMARK 4.2.2. The remaining elements of $(i, j)U_i$ can be neglected. We can immediately proceed to the next coset $(i, j+1)U_i$. The automorphisms found during the traversal of the cosets of U_i provide us with a representative system of $C_{\mathrm{Aut}(\Gamma)}(\{1, \ldots, i-1\})/C_{\mathrm{Aut}(\Gamma)}(\{1, \ldots, i\})$. We want to formulate exactly how to obtain a Sims chain of $\mathrm{Aut}(\Gamma)$:

DEFINITION 4.2.3. For $1 \leq i < n$
$$V_i := C_{\mathrm{Aut}(\Gamma)}(\{1, \ldots, i\}), \quad 1 \leq i < n$$

is the centralizer of the digits $1, \ldots, i$ of $\mathrm{Aut}(\Gamma) =: V_0 =: V$. Then

$$V_i \leq U_i, \quad 0 \leq i < n$$

From our canonicity test, we know the set

$$J(i) := \{j \mid \exists \alpha \in (i,j)U_i \text{ with } \Gamma^\alpha = \Gamma\}$$

We choose for each $j \in J(i)$ an $\alpha_{i,j} \in (i,j)U_i$ with $\Gamma^{\alpha_{i,j}} = \Gamma$. (As mentioned above, we always take the first one we find.) Then the following holds:

THEOREM 4.2.4. *For $1 \leq i < n$, the elements of*

$$\{\alpha_{i,j} \mid j \in J(i)\}$$

form a representative system of the left cosets V_{i-1}/V_i.

PROOF. It remains to show that

$$V_{i-1} = \bigcup_{j \in J(i)} \alpha_{i,j} V_i.$$

The completeness of the decomposition follows from the definition of $J(i)$. We need to demonstrate the disjointness: Let $j, j' \in J(i)$ with $j \neq j'$. Then $(i,j)U_i = \alpha_{i,j}U_i \supseteq \alpha_{i,j}V_i$ (equality holds because $\alpha_{i,j} \in (i,j)U_i$ and containment because $U_i \geq V_i$). Similarly, $\left(i,j'\right)U_i \supseteq \alpha_{i,j}V_i$. Since $(i,j)U_i \cap \left(i,j'\right)U_i = \emptyset$, it follows that $\alpha_{i,j}V_i \cap \alpha_{i,j'}V_i = \emptyset$. □

We obtain the algorithm KATEST() for testing canonicality of a graph. •4.2.2

•4.2.2: **rg:** *A terminer*

Algorithm 5 KATEST Algorithm {algoKatest}

```
1:  Set erz = 0.
2:  for i = n downto 1 do do
3:      for j = i + 1 to n do do
4:          kmn[i] := j
5:          kmn[j] := i
6:          erg = KAREK(i + 1)
7:          kmn[i] := i
8:          kmn[j] := j
9:          if erg = −1 then
10:             return 0
11:         end if
12:     end for
13: end for
```

The following data structures are required:

Algorithm 6 KAREK Algorithm

```
1: procedure KAREK(tz)
2:     if tz = n then
3:         Call ADJVGL().
4:     end if
5:     for each j in J(tz) do
6:         Set α = α_{tz,j}.
7:         if α * π = π then
8:             Continue to the next j.
9:         end if
10:        Recursively call KAREK(tz + 1).
11:    end for
12:    return
13: end procedure
```

Algorithm 7 ADJVGL Algorithm

```
1: procedure ADJVGL
2:     Perform adjacency matrix comparison.
3:     if erg = 0 then
4:         Increment erz by 1.
5:         Store aut[erz].
6:     end if
7: end procedure
```

- An array kmn of length n, which contains the current permutation while traversing the group tree. Before the first call of KATEST(), kmn should be initialized as the identity, i.e., $kmn[i] = i$ for all $i \in \underline{n}$.
- A two-dimensional array aut of size $maxerz \times n$, where the found automorphisms are stored. maxerz should be chosen so that all v_{ij} can be accommodated. We are generous and set maxer $z := \frac{n(n-1)}{2}$.
- A two-dimensional array of size $n \times n$ for the adjacency matrix A. Of course, at this point, another data structure can also be used, depending on the type of objects being considered.
- A global variable erz for the number of found automorphisms, as well as the global variable n and local variables i, j, and erg.

In line 1 of KATEST(), the number of stored automorphisms is set to 0. Line 2 indicates that U_{i-1} is to be processed, and line 3 specifies the coset $(i, j)U_i$. In lines 4 and 5, the transposition (i, j) is

applied. Alternatively, one could write CHANGE $(kmn[i], kmn[j])$, meaning the contents of $kmn[i]$ and $kmn[j]$ are swapped. Line 6 calls KAREK$(i+1)$, which traverses the coset $(i,j)U_i$. The result erg obtained is:

- -1 if an element π was found in $(i,j)U_i$ such that $A \prec A^\pi$, i.e., Γ is not canonical;
- 0 if an automorphism of Γ was found in $(i,j)U_i$;
- 1 otherwise.

In lines 7 and 8, (i,j) is applied again. kmn now contains id again. If the result of KAREK$(i+1)$ was negative, it is clear that Γ is not canonical, and it can be terminated. The return value is then 0. If Γ is canonical, 1 is returned at the end.

4.3. Regarding the function KAREK(tz):

- If the condition in line 1 is satisfied, then a leaf π has been reached in the group tree. The result erg from ADJVGL() is
  -1 if $A \prec A^\pi$, i.e., Γ is not canonical;
  0 if $A = A^\pi$, i.e., π is an automorphism;
  1 otherwise.
  If it is determined that it is an automorphism, we increase the number erz of found automorphisms by 1 and execute the function STORE$(kmn, aut[erz])$. Here, only the contents of kmn are copied to aut[erz]. Line 7 means that we move one step back (upwards) in the group tree.
- If the condition in line 1 is not satisfied, then we are at an inner node in the group tree, more precisely in depth $tz-1$. Then, the for loop starting in line 9 causes all sons in the group tree to be visited one after the other. Line 13 checks whether an automorphism has been found or non-canonicity has been detected and aborts if necessary.

ADJVGL() performs the lexicographical comparison of A and A^π.

EXAMPLE 4.3.1.

Let's consider a concrete example. Let $\Gamma \in \mathcal{R}_{6,3}$ be given by its adjacency matrix A:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

In the following table, we will list for each coset $(i,j)U_i$ traversed during the canonicity test, the first found automorphism, the number of permutations actually tested, and the total number of permutations in each coset (excluding the identity).

Coset	Automorphism	Tested	Total
$(5,6)U_5$	$(5,6)$	1	1
$(4,5)U_4$	-	2	2
$(4,6)U_4$	-	2	2
$(3,4)U_3$	$(3,4)$	1	6
$(3,5)U_3$	-	6	6
$(3,6)U_3$	-	6	6
$(2,3)U_2$	$(2,3)$	1	24
$(2,4)U_2$	$(2,4)$	1	24
$(2,5)U_2$	-	24	24
$(2,6)U_2$	-	24	24
$(1,2)U_1$	$(1,2)(2,5)(3,6)$	91	120
$(1,3)U_1$	$(1,3)(2,5)(3,6)$	91	120
$(1,4)U_1$	$(1,4)(2,5)(3,6)$	91	120
$(1,5)U_1$	$(1,5)$	1	120
$(1,6)U_1$	$(1,6)$	1	120
		343	719

In this example, we can neglect more than half of the permutations of S_6. Particularly important are the savings in the cosets of U_1. Unfortunately, the automorphisms in $(1,2)U_1, (1,3)U_1$, and $(1,4)U_1$ are only found very late, so that only 29 elements can be skipped in each case. Thus, our canonicity test still has some shortcomings.

REMARK 4.3.2. There are situations where one can decide early on that the current branch of the group tree does not need to be traversed:

When we are in depth i of the group tree at node $(1,j_1)\dots(i,j_i) =: \pi$, then all leaves in the subtree of π are elements from πU_i. We denote the upper left submatrix of A^π with i rows and columns as ${A^\pi}_i$. This submatrix remains unchanged when applying $\tau \in U_i$ to A^π: $A^\pi|_i = A^{\pi\tau}|_i$. Often, it can be decided very early based on $A^\pi|_i$ that πU_i can be neglected. The simplest example: $i = 2$ and $A^{(1,j_1)(2,j_2)}(1,2) = 0$. In

the following section, we will work out how to recognize and avoid such situations from the outset. When traversing the group tree in depth i, it is usually not necessary to multiply with all coset representatives of U_{i-1}/U_i.

In these considerations, we return to the normalizers N_i defined in 3.3 and apply canonizing permutations. We make a "canonical pruning of the group tree" (see [Gru1]).

4.4. Canonical Pruning of the Group Tree

We can perform the canonicity test much more efficiently by considering the special structure of the test candidates. In the following, we assume that $\Gamma \in \mathcal{R}_{n,k}$ is semicanonical. Let A be the adjacency matrix of Γ, and let a_i denote the i-th row of A. We use the notations from Section 3.3:

$$\begin{aligned} N_0 &:= S_n = S_{\eta^{(0)}} \\ C_1 &= C_{N_0}(\{1\}) = S_{\zeta^{(1)}} \\ N_i &= N_{C_i}(\Gamma_i) = \left\{\pi \in C_i \mid a_i^\pi = a_i\right\} = S_{\eta^{(i)}}, \\ C_{i+1} &= C_{N_i}(\{1,\ldots,i+1\}) = \left\{\pi \in N_i \mid (i+1)^\pi = i+1\right\} = S_{\zeta^{(i+1)}}, \end{aligned}$$

where $1 \le i < n$ and $\eta^{(0)} = (n)$.

Remark 4.4.1.

There is the following disjoint decomposition of the normalizers N_{i-1} into left cosets of C_i in N_{i-1}:

$$N_{i-1} = \bigcup_{j=i}^{\eta_i^{(i-1)}+i-1} (i,j)C_i, \quad i = 1,\ldots,n$$

We will show in the following that during the traversal of the group tree at depth i, it is not necessary to multiply with all left coset representatives of U_{i-1}/U_i, but only those of C_{i-1}/N_i need to be considered.

Definition 4.4.2.

We define the following sets:

$$\begin{aligned} M_1^{\succ} &:= \left\{\pi \in S_n \mid a_1^\pi \succ a_1\right\}, \\ M_1^{\prec} &:= \left\{\pi \in S_n \mid a_1^\pi \prec a_1\right\}, \\ M_1^{=} &:= \left\{\pi \in S_n \mid a_1^\pi = a_1\right\}. \end{aligned}$$

We have $S_n = M_1^{\succ} \cup M_1^{\prec} \cup M_1^{=}$ and $\mathrm{Aut}(\Gamma) \subseteq M_{=}^{(1)}$. Since $\succ$ compares the matrices row-wise, it is obvious that:

$$\begin{aligned}
&\pi \in M_1^{\succ} \Longrightarrow A^{\pi} \succ A \quad (\Gamma \text{ not canonical }),\\
&\pi \in M_1^{\prec} \Longrightarrow A^{\pi} \prec A,\\
&\pi \in M_1^{=} \Longrightarrow A^{\pi} \succ A \vee A^{\pi} \prec A \vee A^{\pi} = A
\end{aligned}$$

The goal is to describe the sets $M_1^{\succ}, M_1^{\prec}$, and $M_1^{=}$ as simply as possible using the group tree. To do this, we formulate the following.

4.4.0.1. *Lemma:* Let the coset decomposition be given as in 4.1.2:

$$N_0 = \bigcup_{j=1}^{n} (1,j) C_1$$

For each j, let $\sigma_{1,j} \in C_1$ be a canonical permutation of row $a_1^{(1,j)}$. Then:

$$(1,j)\sigma_{1,j} N_1 \subseteq M_1^{=} \text{and } (1,j)\sigma_{1,j}\,(C_1 - N_1) \subseteq M_1^{\prec}$$

Proof. □

Since Γ is regular, we have ${a_1}^{(1,j)\sigma_{1,j}} = a_1$. By definition of N_1, for all $\sigma \in N_1$, we have ${a_1}^{\sigma} = a_1$, and thus ${a_1}^{(1,j)\sigma_{1,j}\sigma} = a_1$ for all $\sigma \in N_1$, so $(1,j)\sigma_{1,j} N_1 \subseteq M_1^{=}$.

Since Γ is semicanonical, for all $\sigma' \in C_1$, we have ${a_1}^{\sigma'} \preceq a_1$, and with the definition of N_1, we obtain ${a_1}^{\sigma} \prec a_1$ for all $\sigma \in C_1 - N_1$. Then we have ${a_1}^{(1,j)\sigma_{1,j}\sigma} = {a_1}^{\sigma} \prec a_1$ for all $\sigma \in C_1 - N_1$, so $(1,j)\sigma_{1,j}\,(C_1 - N_1) \subseteq M_1^{\prec}$.

Remark 4.4.3.

Using the canonical permutation $\sigma_{1,j}$ of row $a_1^{(1,j)}$, we obtain a decomposition of the coset $(1,j)C_1$ into a $M_1^{\prec}$- and a $M_1^{=}$-component. From line 2 onward, we are only interested in elements from

$$M_1^{=} = \bigcup_{j \in \underline{n}} (1,j)\sigma_{1,j} N_1$$

Since these elements fix row 1, we no longer need to consider line 1. We can generalize this approach to the effect that during the traversal of the group tree at depth $i, 1 \le i \le n-2$, we only need to examine line i.

Definition 4.4.4.

For $1 \leq i \leq n-2$, we define the following sets:

$$M_i^{\succ} := \left\{\pi \in S_n \mid {a_j}^{\pi} = a_j, j = 1, \ldots, i-1, {a_i}^{\pi} \succ a_i\right\},$$
$$M_i^{\prec} := \left\{\pi \in S_n \mid {a_j}^{\pi} = a_j, j = 1, \ldots, i-1, {a_i}^{\pi} \prec a_i\right\},$$
$$M_i^{=} := \left\{\pi \in S_n \mid {a_j}^{\pi} = a_j, j = 1, \ldots, i-1, {a_i}^{\pi} = a_i\right\}.$$

THEOREM 4.4.5.

Let $\pi \in M_{i-1}^{=}, (i,j) \in N_{i-1}$, and $\sigma_{i,j} \in C_i$ be a canonical permutation of ${a_i}^{\pi(i,j)}$. Then:

(i) ${a_i}^{\pi(i,j)\sigma_{i,j}} \succ a_i \Longrightarrow \pi(i,j)\sigma_{i,j} \in M_i^{\succ}$ (Γ non-canonical),

(ii) ${a_i}^{\pi(i,j)\sigma_{i,j}} \prec a_i \Longrightarrow \pi(i,j)\sigma_{i,j}C_i \subseteq M_i^{\prec}$,

(iii) ${a_i}^{\pi(i,j)\sigma_{i,j}} = a_i \Longrightarrow \pi(i,j)\sigma_{i,j}N_i \subseteq M_i^{=} \wedge$
$\pi(i,j)\sigma_{i,j}(C_i - N_i) \subseteq M_i^{\prec}$.

PROOF. □

i) Since $\pi \in M_{i-1}^{=}, (i,j) \in N_{i-1}$, and $\sigma_{i,j} \in C_i$, we have ${a_l}^{\pi(i,j)\sigma_{i,j}} = a_l$ for $1 \leq l < i$. Hence, $A^{\pi(i,j)\sigma_{i,j}} \succ A$.

ii) By the definition of the canonical permutation, $\forall \sigma \in C_i : {a_i}^{\pi(i,j)\sigma_{i,j}} \succeq {a_i}^{\pi(i,j)\sigma}$. Therefore, $a_i \succ {a_i}^{\pi(i,j)\sigma_{i,j}} \succeq {a_i}^{\pi(i,j)\sigma} \forall \sigma \in C_i$, which implies $\pi(i,j)C_i = \pi(i,j)\sigma_{i,j}C_i \subseteq M_i^{\prec}$.

iii) According to the definition of N_i, $\forall \sigma \in N_i : {a_i}^{\sigma} = a_i$, thus ${a_i}^{\pi(i,j)\sigma_{i,j}\sigma} = {a_i}^{\sigma} = a_i \forall \sigma \in N_i$, leading to $\pi(i,j)\sigma_{i,j}N_1 \subseteq M_i^{=}$.

Since Γ is semicanonical, $\forall \sigma' \in C_i : a_i \succeq {a_i}^{\sigma'}$, and using the definition of N_i, we get $\forall \sigma \in C_i - N_i : a_i \succ {a_i}^{\sigma}$. Consequently, $a_i \succ {a_i}^{\sigma} = {a_i}^{\pi(i,j)\sigma_{i,j}\sigma} \forall \sigma \in C_i - N_i$, which implies $\pi(i,j)\sigma_{i,j}(C_i - N_i) \subseteq M_i^{\prec}$.

REMARK 4.4.6.

We first apply this theorem to the elements $\pi = (1,j_1)\sigma_{1,j_1}$ and $(2,j_2) \in N_1$. It may well happen that already when examining line 2, we find that A is not canonical (see Example 4.3.10). Particularly effective is also case (ii), where the entire coset $\pi(2,j_2)C_2$ is excluded. In the worst case, we still obtain a split into $M_2^{\prec}$ and $M_2^{=}$ parts. Repeatedly applied, we thus obtain the set

$$M_i^{=} = \bigcup_{\substack{(j_1,\ldots,j_i):\forall l=1,\ldots,i: \\ a_l(1,j_1)\sigma_{1,j_1}\ldots(i,j_l)\sigma_{i,j_l}=a_l}} (1,j_1)\sigma_{1,j_1}\ldots(i,j_i)\sigma_{i,j_i}N_i$$

where for $l = 1, \ldots, i$, $\sigma_{l,j_l} \in C_l$ represents a canonical permutation of the row $a_l^{(1,j_1)\sigma_{1,j_1}\ldots(l-1,j_{l-1})\sigma_{l-1,j_{l-1}}(l,j_l)}$.

Application 4.4.7.

We traverse the group tree as usual using the depth-first method. However, unlike before, we now perform a pruning at depth i according to Theorem 4.3.6. For this, the following row test is to be carried out: Given $\tau \in M^{=}_{(i-1)}$, one must compute the row ${a_i}^{\tau}$ and a canonical permutation σ of row ${a_i}^{\tau}$, as well as evaluate a lexicographic comparison of ${a_i}^{\tau\sigma}$ with a_i. This can be done in one step by sorting the ones of row ${a_i}^{\tau\sigma}$ to the left within the $\zeta^{(i)}$ blocks and checking whether their count matches the count of ones of a_i in the corresponding block.

If situation (i) or (ii) occurs, the test can be aborted, or the current branch in the group tree need not be pursued further. In the worst case (iii), $\tau\sigma N_i$ needs further investigation. Here, we apply the decomposition 4.3.1 of N_i. Thus, $\eta_{i+1}^{(i)}$ children in the group tree are to be considered. If, in the worst-case scenario, we always find equality at Theorem 4.3.6, a total of $\sum_{i=1}^{n-1} \prod_{j=1}^{i} \eta_i^{(i-1)}$ row-wise comparisons must be performed.

Then, of course, a pruning of the group tree can be intensified by finding automorphisms as in Theorem 4.2.1. The two pruning criteria can be easily combined and even "complement" each other to some extent. We will trace the effect of the improved canonicity test using the example from Section 4.2.

Example 4.4.8.

We consider $\Gamma \in \mathcal{R}_{6,3}$ with the adjacency matrix

$$A = \left(\begin{array}{c|c|c|c|c|c} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{array}\right).$$

Here, the $\zeta^{(i)}$ blocks are indicated. We have

$$
\begin{aligned}
N_0 &= S_{(6)} = C_1 \cup (1,2)C_1 \cup (1,3)C_1 \cup (1,4)C_1 \cup (1,5)C_1 \cup (1,6)C_1,\\
N_1 &= S_{(1,3,2)} = C_2 \cup (2,3)C_2 \cup (2,4)C_2,\\
N_2 &= S_{(1,1,2,2)} = C_3 \cup (3,4)C_3,\\
N_3 &= N_4 = S_{(1,1,1,1,2)} = C_5 \cup (5,6)C_5,\\
N_5 &= S_{(1,1,1,1,1,1)} = \{id\}.
\end{aligned}
$$

Thus, at most $6+6*3+6*3*2+6*3*2*1+6*3*2*1*2 = 168$ row tests need to be performed initially. The following table contains a detailed analysis of the refined canonicity test.

Coset	Depth	Tree traversal	Can. perm.	Lex. comparison	
$(5,6)C_5$	5	$(5,6)$	id	$\left({a_5}^{(5,6)} = a_5\right)$	(A)
$(3,4)C_3$	3	$(3,4)$	id	${a_3}^{(3,4)} = a_3$	
	4	$(3,4)$	id	${a_4}^{(3,4)} = a_4$	(A)
$(2,3)C_2$	2	$(2,3)$	id	${a_2}^{(2,3)} = a_2$	
	3	$(2,3)$	id	${a_3}^{(2,3)} = a_3$	
	4	$(2,3)$	id	${a_4}^{(2,4)} = a_4$	(A)
$(2,4)C_2$	2	$(2,4)$	id	${a_2}^{(2,4)} = a_2$	
	3	$(2,4)$	id	${a_3}^{(2,4)} = a_3$	
	4	$(2,4)$	id	${a_4}^{(2,4)} = a_4$	(A)
$(1,2)C_1$	1	$(1,2)$	$(3,6)(4,5)$	${a_1}^{(1,2)(3,6)(4,5)} = a_1$	
	2	$(1,2)(3,6)(4,5)$	id	${a_2}^{(1,2)(3,6)(4,5)} = a_2$	
	3	$(1,2)(3,6)(4,5)$	id	${a_3}^{(1,2)(3,6)(4,5)} = a_3$	
	4	$(1,2)(3,6)(4,5)$	id	${a_4}^{(1,2)(3,6)(4,5)} = a_4$	(A)
$(1,3)C_1$	1	$(1,3)$	$(2,6)(4,5)$	${a_1}^{(1,3)(2,6)(4,5)} = a_1$	
	2	$(1,3)(2,6)(4,5)$	id	${a_2}^{(1,3)(2,6)(4,5)} = a_2$	
	3	$(1,3)(2,6)(4,5)$	id	$a_3(1,3)(2,6)(4,5) = a_3$	
	4	$(1,3)(2,6)(4,5)$	id	${a_4}^{(1,3)(2,6)(4,5)} = a_4$	(A)
$(1,4)C_1$	1	$(1,4)$	$(2,6)(3,5)$	${a_1}^{(1,4)(2,6)(3,5)} = a_1$	
	2	$(1,4)(2,6)(3,5)$	id	${a_2}^{(1,4)(2,6)(3,5)} = a_2$	
	3	$(1,4)(2,6)(3,5)$	id	${a_3}^{(1,4)(2,6)(3,5)} = a_3$	
	4	$(1,4)(2,6)(3,5)$	id	${a_4}^{(1,4)(2,6)(3,5)} = a_4$	(A)
$(1,5)C_1$	1	$(1,5)$	id	${a_1}^{(1,5)} = a_1$	
	2	$(1,5)$	id	${a_2}^{(1,5)} = a_2$	
	3	$(1,5)$	id	${a_3}^{(1,5)} = a_3$	
	4	$(1,5)$	id	${a_4}^{(1,5)} = a_4$	(A)
$(1,6)C_1$	1	$(1,6)$	id	${a_1}^{(1,6)} = a_1$	
	2	$(1,6)$	id	${a_2}^{(1,6)} = a_2$	
	3	$(1,6)$	id	${a_3}^{(1,6)} = a_3$	
	4	$(1,6)$	id	${a_4}^{(1,6)} = a_4$	(A)

It is stated at which depth in the tree traversal which group element is considered. Furthermore, the chosen canonical permutation and the result of the lexicographic comparison are provided. If it is determined that an automorphism has been found, this is indicated by an (A).

In total, 29 row tests are performed. A look at 4.2.6 makes the improvement clear, especially since the old version involves comparing the entire (half) adjacency matrix each time.

For better understanding, it should be noted: As long as only the identity is considered during the tree traversal, no row tests are performed, of course. If the matrices to be compared match up to row 4, then we do not need to compare the last two rows anymore. An automorphism has been found. In depth 1, only the canonical permutation needs to be determined. Equality is then clear due to the regularity.

Figure 4.2 shows the group tree for Γ. The pruning given by the decomposition of the N_i into cosets of C_{i+1} has already been carried out. Thus, the tree has $1+6+6*3+6*3*2+6*3*2*1+6*3*2*1*2 = 169$ nodes (of which $6 * 3 * 2 * 1 * 2 = 72$ are leaves). Nodes where a row test must actually be performed are marked with a "•".

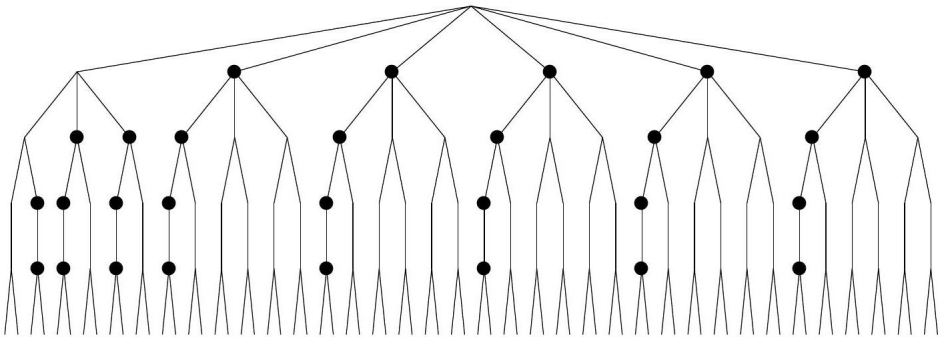


Figure 4.2: A pruned group tree.

EXAMPLE 4.4.9.

Let's calculate our example from 3.5.4:

$$A = \left(\begin{array}{c|c|c|c|c|c|c|c|c} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}\right).$$

It is

$$\begin{aligned} N_0 &= C_1 \cup (1,2)C_1 \cup (1,3)C_1 \cup \ldots \cup (1,9)C_1, \\ N_1 &= C_2 \cup (2,3)C_2 \cup (2,4)C_2 \cup (2,5)C_2, \\ N_2 &= N_3 = C_4 \cup (4,5)C_4, \\ N_4 &= N_5 = C_6 \cup (6,7)C_6, \\ N_7 &= N_8 = \{id\}. \end{aligned}$$

An upper bound for the number of row tests is 9 + 36 + 36 + 72 + 72 + 144 + 144 + 144 = 657. The exact analysis yields the following table:

Coset	Depth	Tree Traversal	Canonical Perm.	Lex. Comparison
$(6,7)C_6$	6	$(6,7)$	id	${a_6}^{(6,7)} = a_6$
	7	$(6,7)$	id	${a_7}^{(6,7)} = a_7$ (A)
$(4,5)C_4$	4	$(4,5)$	$(8,9)$	${a_4}^{(4,5)(8,9)} = a_4$
	5	$(4,5)(8,9)$	id	${a_5}^{(4,5)(8,9)} = a_5$
	6	$(4,5)(8,9)$	id	${a_6}^{(4,5)(8,9)} = a_6$
	7	$(4,5)(8,9)$	id	${a_7}^{(4,5)(8,9)} = a_7$ (A)
$(2,3)C_2$	2	$(2,3)$	id	${a_2}^{(2,3)} \succ a_2$

We have ${a_2}^{(2,3)} = 101110000$. We obtain $\pi = (2,3)$ and edge $(2,4)$ as input for our learning effect from section 3.5.

REMARK 4.4.10.

The canonicality test described so far can be used for any $\Gamma \in \mathcal{G}_n$. Regularity was only used in the proof of Lemma 4.3.3. A more general formulation of 4.3.3, which does not require regularity, can be found in Theorem 4.3.6.

The following statement holds only for regular candidates and allows us to skip entire cosets $(1,i)C_1$ during the canonicality test in some cases.

4.4.0.2. *Lemma:* Let $\Gamma \in \mathcal{R}_{n,k}$. If vertex i is not on a waist circle, then

$$\forall \pi \in (1,i)C_1 : \quad \Gamma^\pi > \Gamma.$$

PROOF. □

The statement follows from Theorem 3.4.1.

REMARK 4.4.11.

Thus, the coset $(1,i)C_1$ only needs to be examined during the canonicality test if vertex i lies on a waist circle. However, for $k > 3$, most vertices lie on a waist circle, so that effective savings are achieved only in the case of cubic graphs. A stronger statement than Theorem 3.4.1 would need to be found in order to increase the efficiency of the canonicality test in this way.

Bibliography

[Bri92] G. Brinkmann, *Generating cubic graphs faster than isomorphism checking*, Universität Bielefeld. SFB 343. Diskrete Strukturen in der Mathematik, 1992.

[GLM97] Thomas Grüner, Reinhard Laue, and Markus Meringer, *Algorithms for group actions: homomorphism principle and orderly generation applied to graphs*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **28** (1997), 113–122.

[Grü95] Thomas Grüner, *Ein neuer ansatz zur rekursiven erzeugung von schlichten graphen*, Master's thesis, Universität Bayreuth (1995).

[Ker91] Adalbert Kerber, *Algebraic combinatorics via finite group actions*, Mannheim etc.: Wissenschaftsverlag, 1991 (English).

[Sim71] Charles C Sims, *Computation with permutation groups*, Proceedings of the second ACM symposium on Symbolic and algebraic manipulation, 1971, pp. 23–28.

4.5. Copyright notice

I confirm that I have written this thesis independently and have only used the aids and sources indicated.

Bayreuth,

8. January 1996