D.M.H van Gent

Graph Isomorphism in Quasi-polynomial Time

Bachelor Thesis

Supervisors:
dr. O. Biesel (MI)
dr. H.J. Hoogeboom (LIACS)

Date Bachelor Exam: 24th June, 2016

LIACS, Universiteit Leiden
Mathematisch Instituut, Universiteit Leiden

Contents

1 Introduction

On the 11th of December 2015, László Babai made public the manuscript for his upcoming paper 'Graph Isomorphism in Quasi-Polynomial Time' on `arXiv.org`, in which he claims to have found an algorithm for deciding whether two graphs are isomorphic in quasi-polynomial time [1]. At the time of this writing, the paper in question has not yet been peer-reviewed. Even so, the manuscript presents new interesting ideas and the build-up to what is actually new would be enough to fill this thesis regardless.

The goal of this thesis is to present parts of Babai's new work, together with previously published results from him and authors like Peter Cameron, Merrick Furst, John Hopcroft, Eugene Luks and Ákos Seress. With this we hope to present an introduction to the to be published paper for a reader with no previous knowledge on the graph isomorphism problem.

We remark that all results in this thesis, unless otherwise stated, are results from Babai's manuscript and are implicitly credited to him. Any proofs of which we could not find a good reference are, although unremarkable, written by the author of this thesis and marked by a dagger (†).

Besides the content of this thesis, we also supply C++ source-code for most of the algorithms we encounter. For this we refer to Appendix B for an overview.

2 Prerequisites

All algorithms in this thesis will concern operations on permutation groups. We assume the reader is familiar with basic group theory and complexity analysis of algorithms. Regardless we present some notation and terminology quickly.

All groups we consider are *permutation groups*, which is a non-empty set of bijections $\Omega \leftrightarrow \Omega$ closed under composition and taking inverses. We write $\mathfrak{S}(\Omega)$ and $\mathfrak{A}(\Omega)$ for the *symmetric* and *alternating group* on the set Ω respectively. We denote subgroups by $H \subseteq G$, leaving implicit that H is also a group. We call $|G|$ and $|\Omega|$ the *order* and *degree* of $G \subseteq \mathfrak{S}(\Omega)$ respectively, and write $[G : H] = |G|/|H|$ for the *index* of H in G. A set $\Delta \subseteq \Omega$ is called *G-stable* if $g(x) \in \Delta$ for all $g \in G$ and $x \in \Delta$. The finest partitioning of Ω into G-stable sets is called the *orbits of G*. We call G *transitive* if it has only one orbit. Given a set $\Delta \subseteq \Omega$, we write $\mathrm{Sstab}_\Delta(G) = \{g \in G \mid g\Delta = \Delta\}$ and $\mathrm{Pstab}_\Delta(G) = \{g \in G \mid (\forall x \in \Delta)\ gx = x\}$ for the *set-wise and point-wise stabiliser of* Δ respectively.

We write $O(f(x)) = \{g : \mathbb{R}_{>0} \to \mathbb{R}_{>0} \mid \limsup_{x\to\infty} g(x)/f(x) \text{ exists}\}$ for the set of functions *asymptotically bounded above by* $f : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$. We also write 2^S and $\binom{S}{t}$ for the *power set* and set of size t subsets of S. For a function $f : A \to X$ and $B \subseteq A$, we write $f|_B : B \to X$ for the *restriction of f to B*.

2.1 Quasi-polynomial Functions

Babai's algorithm is quasi-polynomially bounded in time complexity. Before we can start with anything related to the algorithm, we should first define this

properly.

Definition 2.1. We call $\mathbb{R}[\log(x)]$ the ring of *poly-logarithmic functions* in the indeterminate x. For $f \in \mathbb{R}[\log(x)]$, we call $\exp(f)$ a *quasi-polynomial function.*

Through abuse of notation, we can write the set of quasi-polynomially bounded functions in x as $\exp(\log(x)^{O(1)})$. Note that if f in the above is linear (in $\log(x)$), then $\exp(f)$ is merely a polynomially-bounded function. However, for higher powers, $\exp(f)$ becomes super-polynomial, yet sub-exponential.

Remark 2.2. Like $x^{O(1)}$, the set $\exp(\log(x)^{O(1)})$ is closed under addition, multiplication and composition.

Polynomial algorithms arise from statements of the form 'for all $x \in \Omega$ do'. Logarithms occur when applying divide-and-conquer strategies and walks in binary trees. The factorial appears when working with permutation groups. A natural environment for the somewhat unfamiliar quasi-polynomial complexity class is found through the following:

Lemma 2.3. *For fixed $m \in \mathbb{N}_0$, the function $\log(x)^m!$ is quasi-polynomially bounded in x.*

Proof. Note that $\log(x)^m! \leq \log(x)^{m\log(x)^m} \leq \exp(m\log(x)^{m+1})$, which is quasi-polynomial. □

We could therefore expect quasi-polynomial algorithms to pop up when applying divide-and-conquer strategies to permutation groups. For proving an algorithm has this complexity, we often use the following lemma.

Lemma 2.4 (†)**.** *Let $f, g : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ be weakly increasing functions and let $\alpha \in (0,1)$. If $f(x) \leq f(\alpha x)g(x)$ for all $x > 0$, then $f(x) \leq Mg(x)^{c\log(x)+1}$ for some $M, c \in \mathbb{R}_{>0}$.*

Proof. Let $M = f(1)$ and $c = -1/\log(\alpha)$. Then for all $k \in \mathbb{N}_0$ we have

$$f(x) \leq f(\alpha x)g(x) \leq \ldots \leq f(\alpha^k x)\prod_{i=0}^{k-1} g(\alpha^i x) \leq f(\alpha^k x)g(x)^k.$$

Now pick $k = \lceil c\log(x)\rceil$, then

$$f(\alpha^k x)g(x)^k \leq f(\alpha^{c\log(x)}x)g(x)^{c\log(x)+1} \leq f(1)g(x)^{c\log(x)+1} \leq Mg(x)^{c\log(x)+1}$$

for all $x \in \mathbb{R}_{>0}$. □

Corollary 2.5. *Let $f, g : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ be weakly increasing functions and let $\alpha \in (0,1)$. If $f(x) \leq f(\alpha x)g(x)$ for all $x > 0$ and g is quasi-polynomially bounded, then so is f.*

This g occurs in the form of 'multiplicative cost'. In divide-and-conquer strategies, one reduces an instance of the problem to a number of smaller ones. This reduction itself costs some operations, which we call additive costs. The number of instances we reduce to is the multiplicative cost. The motivation behind this name becomes clear when we take a complexity function T. Then $T(x) = q(x)T(\alpha x) + f(x)$, where q and f are the multiplicative and additive cost respectively, and α the reduction factor.

2.2 The Reduction of Graph Isomorphism to String Isomorphism

Instead of solving the graph isomorphism problem directly, we solve the string isomorphism problem to which graph isomorphism polynomially reduces.

Definition 2.6. A *string* is a map $\mathfrak{x} : \Omega \to \Sigma$, where $|\Omega| \in \mathbb{N}_1$. We call Ω the *places* and Σ the *alphabet*.

Definition 2.7. A bijection $\sigma : \Omega \leftrightarrow \Omega'$ is an *isomorphism* of $\mathfrak{x} : \Omega \to \Sigma$ and $\mathfrak{y} : \Omega' \to \Sigma$ if $\mathfrak{x} \circ \sigma^{-1} = \mathfrak{y}$. We write $\sigma(\mathfrak{x}) = \mathfrak{x} \circ \sigma^{-1}$, $\mathrm{Iso}(\mathfrak{x}, \mathfrak{y})$ for the set of isomorphisms, and $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) := G \cap \mathrm{Iso}(\mathfrak{x}, \mathfrak{y})$ for any G.

Example 2.8. We intuitively write $\mathfrak{x} = abcd$ and $\mathfrak{y} = bcda$ for strings $\mathfrak{x}, \mathfrak{y} : \{1, 2, 3, 4\} \to \{a, b, c, d\}$. Then in cycle-notation $(1\ 4\ 3\ 2)\mathfrak{x} = \mathfrak{y}$.

Consider the following problems:

Graph Isomorphism (GI):
Given graphs $X = (\Omega, E)$ and $Y = (\Omega, F)$, determine $\mathrm{Iso}(X, Y)$.

String Isomorphism (SI):
Given strings $\mathfrak{x}, \mathfrak{y} : \Omega \to \Sigma$ and group $G \subseteq \mathfrak{S}(\Omega)$, determine $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$.

The complexity of the above problems is measured in $|\Omega|$, which from here on will be called n. Since $|\mathfrak{S}(\Omega)| = n!$, we cannot, in general, store all elements of a group in memory without exceeding a quasi-polynomial limit of operations in n, let alone polynomial. All groups and cosets we consider are therefore represented by a list of m generators, and the complexity of our algorithms should also take into account this m. However, at cost polynomial in m, we can reduce the number of generators to at most n^2 (Theorem 2.11). Because of this, we can assume $m \in O(n^2)$ and disregard the contribution of m to the complexity of the algorithms. Note that membership testing in groups becomes non-trivial when considering only generators of a group. More on this in Subsection 2.3.

Lemma 2.9. *If SI is $O(f(n))$, then GI is $O(f(n^2) + n^2)$.*

Proof. Let $X = (\Omega, E)$ and $Y = (\Omega, F)$ be given. Define $\mathfrak{x}, \mathfrak{y} : \Omega^2 \to \{0, 1\}$ as the indicator functions $\mathfrak{x} = \mathbb{1}_E$ and $\mathfrak{y} = \mathbb{1}_F$. Consider the natural action $S = \mathrm{im}(\mathfrak{S}(\Omega) \to \mathfrak{S}(\Omega^2))$ of $\mathfrak{S}(\Omega)$ on Ω^2, and determine using SI the group $I = \mathrm{Iso}_S(\mathfrak{x}, \mathfrak{y})$. Then $g \in I$ if and only if $(e_1, e_2) \in E \Leftrightarrow (ge_1, ge_2) \in F$, so $I = \mathrm{Iso}(X, Y)$. The complexity claim follows trivially. □

Babai's actual result is the following.

Theorem 2.10 (Babai)**.** *SI is in* $\exp(\log(n)^{O(1)})$.

From this point on the reader is allowed to forget anything they know about the graph isomorphism problem.

2.3 Basic Group Theoretic Algorithms

Before we can start properly, we need polynomial algorithms for some basic group operations, which we will present in this section.

Theorem 2.11 (Furst, Hopfcroft, Luks [2])**.** *Let $G = \langle g_1, \ldots, g_m \rangle \subseteq \mathfrak{S}_n$ be a group represented by its generators and let $1 = G_0 \subseteq \ldots \subseteq G_k = G$ be a chain of subgroups where the G_i with $i < k$ are represented by a $O(f(n))$-time computable membership-testing function and $[G_{i+1} : G_i] \in O(g(n))$. Then one can determine in $O((k^2 g(n)^2 + m)kf(n)g(n))$ time:*

1. *the order of a G_i;*
2. *whether $\sigma \in \mathfrak{S}_n$ is an element of G;*
3. *generators for a G_i and coset representatives of G_{i+1}/G_i.*

Proof. We do not present the entire proof, since a great reference is available. The idea is that we create the set of coset representatives C_{i+1} of G_{i+1}/G_i for each i. We do this by taking a $g \in G$ and finding a $c_i \in C_i$ for each i such that $g = c_1 \cdots c_k$, which we call filtering. If we can't find such c_i at some point we add $c_k^{-1} \cdots c_{i+1}^{-1} g$ to C_i. It turns out it is sufficient to filter all $g \in \{g_1, \ldots, g_m\}$ and then all $g \in C_i C_j$ until the C_i become stable.

Using basic group theory, generators (3) for G_i are just $C_1 \cup \cdots \cup C_i$, and the order (1) is $|C_1| \cdots |C_i|$. For membership testing (2) of $g \in \mathfrak{S}_n$, we filter g and find $c_i \in C_i$ such that $g = c_1 \cdots c_k$ if and only if $g \in G$. □

Remark 2.12. For any $G \subseteq \mathfrak{S}_n$, we can pick the point-wise stabilisers $G \supseteq \mathrm{Pstab}_{\{1\}}(G) \supseteq \mathrm{Pstab}_{\{1,2\}}(G) \supseteq \ldots \supseteq 1$ as chain of subgroups, since they have index $\leq n$ and membership is testable in constant time. This degenerate case admits some optimizations in which we lose the factor $g(n)$, and an implementation of this algorithm is provided.

Corollary 2.13. *Let $\phi : G \to H$ be a homomorphism that is computable in polynomial time and let $I \subseteq H$ be a subgroup such that $[G : \phi^{-1}(I)] \in n^{O(1)}$, then we can find generators for $\phi^{-1}(I)$ in polynomial time using the above algorithm, since membership to $\phi^{-1}(I)$ is testable through ϕ in polynomial time.*

Lemma 2.14 (†)**.** *Let $G \subseteq \mathfrak{S}_n$ and H be groups, and let $\phi : G \to H$ be a surjective homomorphism that is computable in $n^{O(1)}$ time, defined on the generators $\langle g_1, \ldots, g_m \rangle = G$ and let $h \in H$. Then we can find a pull-back $g \in \phi^{-1}(h)$ in $n^{O(1)}$ time.*

Proof. We apply an extended version of Theorem 2.11 to $\langle \phi(g_1), \ldots, \phi(g_m) \rangle$. Append to the algorithm maps $f_i : C_i \to G$, in which we store $f_i(\phi(g)) = g$ for each $\phi(g)$ we filter. Then for each element $h \in C_i C_j$ we filter subsequently, we know a pull-back. With minimal bookkeeping we can then update the f_i with a pull-back for the filtrate. Then all coset representatives have a pull-back associated with them, which can be retrieved when doing an H membership test. Again an implementation is provided. □

Another algorithm we need is one for finding block-systems in transitive groups. For this we need some definitions first.

Definition 2.15. Let $N \subseteq \mathfrak{S}(\Gamma)$ and $G \subseteq \mathfrak{S}_p$ for some $r \in \mathbb{N}_{\geq 1}$ be permutation groups. Then the *imprimitive wreath product of N and G* is

$$N \wr G := \{(\gamma, i) \mapsto (n_{gi}\gamma, gi) \mid (n_i)_i \in N^p, g \in G\} \subseteq \mathfrak{S}(\Gamma \times \{1, \dots, p\}).$$

Whenever we have disjoint sets $P_1, \dots, P_p$ of equal size, we often write $\mathfrak{A}(P_i) \wr_i \mathfrak{S}_p$ or $\mathfrak{S}(P_i) \wr_i \mathfrak{S}_p$ for the obvious subgroups of $\mathfrak{S}(P_1 \sqcup \dots \sqcup P_p)$.

Definition 2.16. Let $G \subseteq \mathfrak{S}(\Omega)$ be a group and let $P_1 \sqcup \ldots \sqcup P_m = \Omega$ be a partitioning. If for all $g \in G$ and i there exists a j such that $gP_i = P_j$, we say that G *respects the partitioning* $\{P_i\}_i$ and we call $\{P_i\}_i$ a *block-system* for G. The *trivial block-systems* are $\{\Omega\}$ and $\{\{\omega\} \mid \omega \in \Omega\}$, and we call G *primitive* if it admits only trivial block-systems. A block-system is *maximal* if it is not $\{\Omega\}$ and cannot be made coarser into a block-system other than $\{\Omega\}$.

Remark 2.17. If G is transitive, then the blocks in a block-system are necessarily of equal size. Then saying that $P_1 \sqcup \dots \sqcup P_p$ is a block-system of G is equivalent to saying that $G \subseteq \mathfrak{S}(P_i) \wr_i \mathfrak{S}_r$. Also note that maximal block-systems are not unique. For example, the cyclic group C_6 has maximal block-systems $\{1,4\} \sqcup \{2,5\} \sqcup \{3,6\}$ and $\{1,3,5\} \sqcup \{2,4,6\}$, which become evident when we write C_6 as $C_2 \times C_3$.

Theorem 2.18 (Atkinson [3])**.** *Given a group $\langle g_1, \dots, g_m \rangle \subseteq \mathfrak{S}(\Omega)$, we can find a maximal block-system in $O(n^2 \log_2(n) m \alpha(n))$ time, where α is the inverse of $A(n,n)$ with A the Ackermann function. (For all practical purposes, $\alpha(n) \leq 5$.)*

Proof. Again we present no proof due to the page limit of this thesis, but a good though slightly old reference exists. We note that the paper describes an $O(n^2 m \log(n))$ algorithm for finding minimal block-systems, which we apply at most $\log_2(n)$ times iteratively to obtain maximal block-systems. The $n \log(n)$ in $O(nm \log(n))$ improves to $n\alpha(n)$ when we replace its main data-structure by a union-find [4]. We provide an implementation. □

Now that we can find block-systems, we also need some algorithms to work with them.

Definition 2.19. Let $\phi : A \to B$ be a function and $B' \subseteq B$ be a subset. Then a subset $A' \subseteq A$ is called an *improper pull-back of B' in ϕ* if $\phi(A') = B'$.

Remark 2.20. In the case ϕ is an injective function, the improper pull-back is necessarily $\phi^{-1}(B')$, called the *(proper) pull-back*. Through Lemma 2.14, we can always find an improper pull-back of a subgroup in polynomial time by finding a pull-back for each of the generators of B'.

In the special case where ϕ is a projection onto a quotient group, we do not need the index of the subgroup to be bounded as in Corollary 2.13. We only require generators for the divisor group.

Lemma 2.21. *Given $G \subseteq \mathfrak{S}(\Omega)$ and block-system $\mathcal{P} = \{P_1, \dots, P_p\}$ of Ω, we can find $\mathrm{Sstab}_{P_p}(G) =: H$ in polynomial time.*

Proof. Again apply Theorem 2.11 as in Remark 2.12. That is, using the chain $1 \subseteq \cdots \subseteq \mathrm{Pstab}_{\{\omega_1,\omega_2\}}(H) \subseteq \mathrm{Pstab}_{\{\omega_1\}}(H) \subseteq H \subseteq G$ after choice of some arbitrary ordering $\{\omega_1, \ldots, \omega_n\} = \Omega$. Note that

$$\begin{aligned}[G : H] &= [G \cap \mathfrak{S}(P_i) \wr_i \mathfrak{S}_p : G \cap \mathfrak{S}(P_p) \times (\mathfrak{S}(P_i) \wr_i \mathfrak{S}_{p-1})] \\ &\leq [\mathfrak{S}(P_i) \wr_i \mathfrak{S}_p : \mathfrak{S}(P_p) \times (\mathfrak{S}(P_i) \wr_i \mathfrak{S}_{p-1})] = p,\end{aligned}$$

and is thus linearly bounded in n, so the algorithm is polynomial. □

Lemma 2.22. *Let $G \subseteq \mathfrak{S}(\Omega)$ be a group with block-system $\mathcal{P} = \{P_1, \cdots, P_p\}$, and let $\pi : G \to \mathfrak{S}(\mathcal{P})$ be the induced action on the blocks. Given generators for any $I \subseteq \mathrm{im}(\pi)$, we can find generators for the pull-back $\pi^{-1}(I)$ in polynomial time.*

Proof. Let $H = \bigcap_{i=1}^{p} \mathrm{Sstab}_{P_i}(G)$ the set-wise stabiliser of all blocks. We can find H by iteratively applying Lemma 2.21 to G for each of the blocks. Then $\pi^{-1}(I) = \langle H, X \rangle$, where X is an improper pull-back of I. □

We could also, for efficiency, apply Theorem 2.11 once to the chain $1 \subseteq H \subseteq \bigcap_{i=1}^{p-1} \mathrm{Sstab}_{P_i}(G) \subseteq \cdots \subseteq \bigcap_{i=1}^{1} \mathrm{Sstab}_{P_i}(G) \subseteq G$.

Lemma 2.23. *Let $G \subseteq \mathfrak{S}(\Omega)$ and let $\Delta \subseteq \Omega$ be G-invariant with projection map $\pi_\Delta : G \to \mathfrak{S}(\Delta)$. Given $I \subseteq \mathrm{im}(\pi_\Delta)$, we can find the pull-back $\pi_\Delta^{-1}(I)$ in polynomial time.*

Proof. Again it is sufficient to find generators for $\mathrm{Pstab}_\Delta(G)$ using Remark 2.12. □

With all this we are sufficiently equipped to properly start with the string isomorphism problem.

2.4 Luks' String Isomorphism Algorithm

In this subsection we introduce the base algorithm used by Luks in his treatment of SI for a special class of groups [5]. We will see that this algorithm does not run in polynomial time in the general case, and identify what is needed to extend the algorithm to a quasi-polynomial one. Unlike of the rest of this thesis, this subsection is entirely due to Luks.

Assuming we can find isomorphisms in G, we extend our capabilities of solving SI to cosets using

$$\mathrm{Iso}_{\sigma G}(\mathfrak{x}, \mathfrak{y}) = \{\sigma g \in \sigma G \mid \sigma g \mathfrak{x} = \mathfrak{y}\} = \sigma \,\mathrm{Iso}_G(\mathfrak{x}, \sigma^{-1}\mathfrak{y}). \qquad \text{(Shift identity)}$$

When $\Delta \subseteq \Omega$ is G-invariant, we get using Lemma 2.23 that

$$\begin{aligned}\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) &= \pi_\Delta^{-1} \mathrm{Iso}_{\pi_\Delta G}(\mathfrak{x}|_\Delta, \mathfrak{y}|_\Delta) \cap \pi_{\Delta^c}^{-1} \mathrm{Iso}_{\pi_{\Delta^c} G}(\mathfrak{x}|_{\Delta^c}, \mathfrak{y}|_{\Delta^c}) \\ &= \pi_{\Delta^c}^{-1} \mathrm{Iso}_{\pi_{\Delta^c} \pi_\Delta^{-1} \mathrm{Iso}_{\pi_\Delta G}(\mathfrak{x}|_\Delta, \mathfrak{y}|_\Delta)}(\mathfrak{x}|_{\Delta^c}, \mathfrak{y}|_{\Delta^c}).\end{aligned} \qquad \text{(Chain rule)}$$

Note that in the last equality π_{Δ^c} is a map $\pi_\Delta^{-1} \mathrm{Iso}_{\pi_\Delta G}(\mathfrak{x}|_\Delta, \mathfrak{y}|_\Delta) \to \mathfrak{S}(\Delta^c)$ instead of $G \to \mathfrak{S}(\Delta^c)$. The above works because $\mathrm{Iso}_{\pi_\Delta G}(\mathfrak{x}|_\Delta, \mathfrak{y}|_\Delta)$ is either empty or a coset of $\mathrm{Aut}_{\pi_\Delta G}(\mathfrak{x}|_\Delta)$.

With the chain rule we reduce the problem to transitive groups. We can even generalise this concept from G-stable sets to block-systems.

Consider a subgroup $H \subseteq G$. Using the shift identity we get

$$\mathrm{Iso}_G(\mathfrak{x},\mathfrak{y}) = \mathrm{Iso}_{\bigcup_{\sigma H \in G/H} \sigma H}(\mathfrak{x},\mathfrak{y}) = \bigcup_{\sigma H \in G/H} \sigma \,\mathrm{Iso}_H(\mathfrak{x},\sigma^{-1}\mathfrak{y}). \qquad \text{(Weak Luks reduction)}$$

Now finding Iso_G reduces to $[G:H]$ times finding Iso_H. The only caveat is that we need to compute the union of these sets.

Lemma 2.24. *Given each* $\mathrm{Iso}_H(\mathfrak{x},\sigma^{-1}\mathfrak{y})$ *with* $\sigma H \in G/H$*, we can compute* $\mathrm{Iso}_G(\mathfrak{x},\mathfrak{y})$ *in* $O([G:H]\cdot n)$ *time.*

Proof. Enumerate the non-empty $\sigma_i I_i := \sigma_i \,\mathrm{Iso}_H(\mathfrak{x},\sigma_i^{-1}\mathfrak{y})$ by $1 \le i \le m$ for $\sigma_i H \in G/H$. If all are empty, $\mathrm{Iso}_G(\mathfrak{x},\mathfrak{y}) = \emptyset$. Otherwise, write $\sigma_i I_i = c_i \,\mathrm{Aut}_H(\mathfrak{x})$ and $d_i = c_m^{-1} c_i$ for all $i \le m$. Now consider $J = \langle d_1,\dots,d_{m-1},\mathrm{Aut}_H(\mathfrak{x})\rangle$. Clearly each element of J is a G-automorphism of $\mathfrak{x}$, and thus $c_m J \subseteq \mathrm{Iso}_G(\mathfrak{x},\mathfrak{y})$. Conversely, $\sigma_i I_i = c_m d_i \,\mathrm{Aut}_H(\mathfrak{x}) \subseteq c_m J$ for all i. We conclude that $\mathrm{Iso}_G(\mathfrak{x},\mathfrak{y}) = c_m J$. □

Now, if we have a block system $\mathcal{P}$, we can let H be its set-wise stabiliser, such that the blocks become orbits. If we combine this with weak Luks reduction and the chain rule, we reduce these transitive groups as well. This is called *strong Luks reduction*. The result is the following:

Algorithm 2.25. `LuksStringIsomorphism`

input : A permutation group $G \subseteq \mathfrak{S}(\Omega)$ and strings $\mathfrak{x},\mathfrak{y} : \Omega \to \Sigma$.
output: $\mathrm{Iso}_G(\mathfrak{x},\mathfrak{y})$

1 **if** $G \subseteq \mathrm{Aut}(\mathfrak{x})$ **then** (Recursion base)
2 | **if** $\mathfrak{x} = \mathfrak{y}$ **then return** G;
3 | **else return** $\emptyset$;
4 Find maximal block-system $\mathcal{P} = \{P_1,\dots,P_p\}$ of G ; `/* Thm 2.18 */`
5 Determine set-wise stabiliser $H \subseteq G$ of the blocks ; `/* Lem 2.21 */`
6 Let C be a set of coset representatives of G/H;
7 $R \leftarrow \emptyset$;
8 **foreach** $\sigma \in C$ **do** (Strong Luks reduction)
9 | $\tau \leftarrow \sigma$, $I \leftarrow H$;
10 | **foreach** $P \in \{P_1,\dots,P_p\}$ **do** (Chain rule)
11 | | Let $\pi : I \to \mathfrak{S}(P)$ be the restriction to the block;
12 | | $\tau I \leftarrow \tau\pi^{-1}$(`LuksStringIsomorphism`$(\pi(I), \mathfrak{x}|_P, (\tau^{-1}\mathfrak{y})|_P)$);
13 | $R \leftarrow R \cup \tau I$; `/* Lem 2.24 */`
14 **return** R;

An implementation of this algorithm is provided.

Luks described this algorithm when solving GI for graphs of bounded degree d. Using this restriction he considered a special case of SI, where he could bound the index $[G:H]$ by a constant depending on d. This results in a constant multiplicative cost in the recursive step, and in turn this gives a polynomial

algorithm. It is clear that in the general case, this algorithm becomes super-polynomial. Take for example the primitive group $G = \mathfrak{S}(\Omega)$ with χ non-constant. Then the maximal block-system we find are the the singletons in Ω, and its stabiliser is the trivial group. It follows that $C = \mathfrak{S}(\Omega)$ and thus the loop on line 8 repeats $n!$ times.

A quasi-polynomially bounded index is enough to make the algorithm quasi-polynomial by Corollary 2.5. It remains to solve the case of groups 'primitive enough' to exceed such bound.

3 Canonicity

In this thesis we consider a number of categories with which we need to become familiar. Furthermore, in this section we present some concepts from category theory that will help in describing the coming algorithms. Since we are only interested in isomorphisms, we define our categories such that they are the only morphisms. We remark that broader definitions of the following categories exist, that admit non-bijective morphisms as well. A non-exhaustive list is as follows:

`Str`: the category of *strings* (Definitions 2.6 and 2.7).

`SGph`: the category of *simple graphs.*
The objects are (V, E) with V a finite set and $E \subseteq \binom{V}{2}$. The morphisms are bijections $f : V \leftrightarrow V'$ such that $\{a, b\} \in E \Leftrightarrow \{f(a), f(b)\} \in E'$.

`BiGph`: the category of *bipartite simple graphs.*
The objects are $(V_1, V_2; E)$ with V_1, V_2 finite disjoint and $E \subseteq V_1 \times V_2$. The morphisms are bijections $f : V_1 \sqcup V_2 \leftrightarrow V_1' \sqcup V_2'$ such that $f(V_i) = V_i'$ and $(a, b) \in E \Leftrightarrow (f(a), f(b)) \in E'$.

`ColEq`: the category of *coloured equipartitions (on finite sets).*
The objects are $(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ where the $\mathcal{P}_i$ partition disjoint finite sets, and $A, B \in \mathcal{P}_i \Rightarrow |A| = |B|$. The morphisms are bijections $f : \bigsqcup_i \bigsqcup_{P \in \mathcal{P}_i} P \leftrightarrow \bigsqcup_i \bigsqcup_{P' \in \mathcal{P}_i'} P'$ such that $a, b \in P \in \mathcal{P}_i \Leftrightarrow (\exists P')(f(a), f(b) \in P' \in \mathcal{P}_i')$.

`CoCnf`: the category of *coherent configurations* (Definitions 6.8 and 6.3).

All categories $\mathcal{C}$ we consider come equipped with a faithful functor $\square : \mathcal{C} \to$ `FinSet` we call the *underlying (finite) set*, meaning that it is injective on morphisms between two given objects. For `Str` the underlying sets are the places and for graphs the vertex sets. Often the objects under consideration all share the same underlying set, which we then denote by Ω. In these categories the only morphisms we consider are isomorphisms, which makes the categories groupoids. As a consequence of the functorial properties of $\square$, the endomorphisms in $\mathcal{C}$ induce bijections on the underlying set. The set of endomorphisms of an object from $\mathcal{C}$ can therefore be interpreted as a permutation group on Ω. For this reason it makes sense to speak of *automorphism groups* and *isomorphism cosets* in these categories. Since we make this identification with permutation groups, we make no distinction between morphisms of different categories that induce the same permutation.

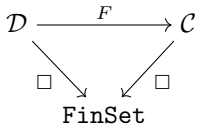
3.1 Relative Canonicity

In the coming sections we describe constructions of an object B from another object A, like the construction of a graph from a coherent configuration. These constructions are always *canonical* in the sense that isomorphic objects A give rise to isomorphic objects B. This is the same as saying that the construction is a functor between the categories of A and B. We will describe a more nuanced concept of canonicity, called relative canonicity.

Example 3.1. Consider the category $\mathtt{SGph}_\Omega$ of graphs with underlying set $\Omega = \{1, \ldots, n\}$, and let $G, H \in \mathrm{obj}(\mathcal{C})$. Furthermore, let $\mathtt{SGphP}_\Omega$ be the category of such graphs equipped with a special point that needs to be preserved by morphisms. If we define $F_a : \mathtt{SGph}_\Omega \to \mathtt{SGphP}_\Omega$ that assigns $a \in \Omega$ as special point to a graph, then this is not a functor if $n > 1$. Morphisms in $\mathtt{SGph}_\Omega$ can move a to a point that is not a, an operation that does not commute with F_a. However, F_a is 'almost canonical'. Namely $\bigsqcup_{a\in\Omega} \mathrm{Iso}_{\mathtt{SGphP}}(F_1(G), F_a(H)) = \mathrm{Iso}_{\mathtt{SGph}}(G, H)$, since any graph isomorphism must map 1 to precisely one point.

We make this concept more rigorous with the following definition.

Definition 3.2. A *reduction from category $\mathcal{C}$ to $\mathcal{D}$* is a functor $F : \mathcal{D} \to \mathcal{C}$, surjective on objects, such that the triangle



commutes, and for every $f \in \mathrm{Iso}_{\mathcal{C}}(A, B)$ and $D \in F^{-1}(A)$ there exists a unique $E \in \mathrm{obj}(\mathcal{D})$ and $g \in \mathrm{Iso}_{\mathcal{D}}(D, E)$ such that $F(g) = f$. For any $A \in \mathrm{obj}(\mathcal{C})$ we call $|F^{-1}(A)|$ the *multiplicative cost of the reduction of A*.

Lemma 3.3. *If $F : \mathcal{D} \to \mathcal{C}$ is a reduction, then*

$$\mathrm{Iso}_{\mathcal{C}}(A, B) = F \bigsqcup_{E \in F^{-1}(B)} \mathrm{Iso}_{\mathcal{D}}(D, E) \quad \textit{for all } D \in F^{-1}(A).$$

Proof. Since F is a functor, we clearly have that $F(g) \in \mathrm{Iso}_{\mathcal{C}}(A, B)$ for all $g \in \mathrm{Iso}(D, E)$ such that $F(D) = A$ and $F(E) = B$, so the inclusion '$\supseteq$' holds. Conversely, for each $f \in \mathrm{Iso}_{\mathcal{C}}(A, B)$ there exist objects $D \in F^{-1}(A)$ and $E \in F^{-1}(B)$ since F is surjective on objects, and a map $g \in \mathrm{Iso}_{\mathcal{D}}(D, E)$ such that $F(g) = f$, so the inclusion '$\subseteq$' holds as well. By the uniqueness, the union is disjoint. □

The construction of D, E from A, B in the above is called *canonical with respect to choice of E* or *canonical relative to E*. These reductions will play the role of division in the coming divide-and-conquer algorithms. Note that it follows immediately from Definition 3.2 that the composition of reductions is a reduction.

Example 3.4. In Example 3.1, the functor we would construct for a proper reduction would be the forgetful functor from $\mathtt{SGphP}_\Omega$ to $\mathtt{SGph}_\Omega$. Note that Weak Luks reduction is a reduction at multiplicative cost $[G : H]$ as well, when we consider the allowed isomorphisms σH in $\mathrm{Iso}_{\sigma H}(\mathfrak{r}, \mathfrak{y})$ to be a property of the objects instead of the morphisms.

We also need to consider the computational aspect of reduction at multiplicative cost. We can say an algorithm reduces an $A \in \mathrm{obj}(\mathcal{C})$ at multiplicative cost if it returns all objects in $F^{-1}(A)$ as in the above. Subjectively, this becomes rather unwieldy and unreadable when we want to compose such reductions and apply recursive strategies to them. Instead, we define it as follows:

Definition 3.5. We say an algorithm $\mathtt{X}$ reduces input $A, B \in \mathrm{obj}(\mathcal{C})$ at multiplicative cost $q(n)$ if it also takes an algorithm $\mathtt{Y}$ as input that computes $\mathrm{Iso}_{\mathcal{D}}(A', B')$ for any $A', B' \in \mathrm{obj}(\mathcal{D})$ with the same underlying sets as A, B, which $\mathtt{X}$ calls at most $q(n)$ times to compute $\mathrm{Iso}_{\mathcal{C}}(A, B)$.

In this thesis and the code provided we will use anonymous functions, also called lambda abstractions, to construct and pass around our algorithm $\mathtt{Y}$ to $\mathtt{X}$. This is syntactic sugar supported by many old and modern programming languages, for example Haskell, Lisp, C++ and Java. A property they have we make liberal use of is their ability to capture any variable in the scope of their definition by reference automatically.

Example 3.6. Continuing Example 3.1, we construct a simple reduction at multiplicative cost $|V'|$, called $\mathtt{X}$:

input : Simple graphs $G = (V, E)$ and $G' = (V', E')$ and an algorithm $\mathtt{Y}$ that computes the isomorphisms of pointed graphs.
output: $\mathrm{Iso}(G, G')$.

$I \leftarrow \emptyset$;
Take any $x \in V$ and let $G_x = (V, x, E)$ be its pointed graph;
foreach $y \in V'$ **do**
  $G'_y \leftarrow (V', y, E')$;
  $I \leftarrow I \cup \mathtt{Y}(G_x, G'_y)$;
return I;

We could then call $\mathtt{X}(G, G', \mathtt{Y})$ with for example the function $\mathtt{Y}$ that computes the isomorphisms by brute force.

4 Cameron Groups

Our current objective is to deal with the groups that Luks' algorithm cannot handle in quasi-polynomial time. In this case we have a block-system $\mathcal{P}$ on which G acts as a primitive group, since $\mathcal{P}$ is a coarsest block-system. For now we only consider the action on this block system, and thus treat the blocks as points and G as the permutation group on those points. In this chapter we characterise the groups that can arise and present algorithmic tools to uncover their structure. First we will define a class of groups.

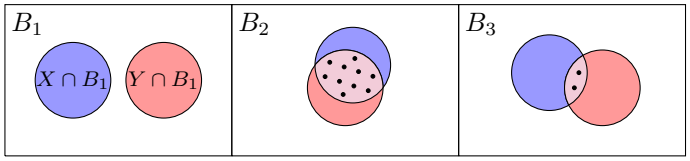


Figure 1: Visual representation of two transversals intersecting.

Definition 4.1. Let $k, r, t \in \mathbb{N}_1$ be parameters. Let B be a set of size k and let $C = B \times \{1, \ldots, r\} =: rB$ be the disjoint union of r copies of B, with $B_i := B \times \{i\}$. Then $A = \{X \subseteq C \mid (\forall i)|X \cap B_i| = t\}$ is the set of *t-transversals in C*.

Definition 4.2. Given t-transversals X, Y of $C = rB$, the *intersection pattern* is the sorted sequence $f_1 \leq \ldots \leq f_r$ of *intersection numbers* $d_i := |B_i \cap X \cap Y|$.

Example 4.3. In Figure 1 we have $k = 120$, $t = 20$ and $r = 3$. We consider two transversals $X, Y \in A$ in blue and red with intersection numbers $0, 10, 2$ and intersection pattern $0, 2, 10$.

In the language of the coming sections, the t-transversals form a binary relational structure on A with relations $R_p \subseteq A^2$ that contains all pairs with a given intersection pattern p. Note that the relations can be canonically ordered, for the example using the lexicographical order on intersection patterns. Such a structure is called a *Cameron scheme*, where a scheme is a homogeneous symmetric binary coherent configuration. The following group turns out to be interesting:

Definition 4.4. Let A be the set of t-transversals on the set $C = rB$, with $|B| \geq 2t$. Let H be such that $\mathfrak{A}(B_1) \times \cdots \times \mathfrak{A}(B_r) \subseteq H \subseteq \mathfrak{S}(B_i) \wr_i \mathfrak{S}_r$. Then the image $G = \mathrm{im}(H \to \mathfrak{S}(A))$ of the natural H-action on A is called a *Cameron group*.

Note that the $\mathfrak{A}(\Omega)$ and $\mathfrak{S}(\Omega)$ are themselves Cameron groups, and that the B_i form a block-system for H in the definition. That there are primitive groups with super-quasi-polynomial order with respect to their degree comes as no surprise (take for example $\mathfrak{S}(\Omega)$). What does, however, is that Cameron groups and some sporadic groups turn out to be the only ones.

Theorem 4.5 (Cameron [6], Maróti [7])**.** *Let $G \subseteq \mathfrak{S}(\Omega)$ with $|\Omega| = n \geq 25$ be a primitive group of order $|G| \geq n^{1+\log_2 n}$. Then G is a Cameron group.*

Note that given a Cameron group, the identification of Ω with t-transversals containing points from some $C = rB$ is a priori unknown. An older result of Babai, Luks and Seress [8] is that we can recover this structure in $O(\log(n))$ time on $n^{O(1)}$ processors. Here it is enough that this implies the algorithm is polynomial on a single processor. This identification comes through the following.

Definition 4.6. Let $G \subseteq \mathfrak{S}(\Omega)$ act naturally on some G-stable subset $A \subseteq 2^\Omega$ such that $\bigcap_{X : x \in X \in A} X = \{x\}$ for all $x \in \Omega$. Then for $x \in \Omega$, we define its *A-dual point* as $\mathrm{dual}_A(x) := \{X \mid x \in X \in A\}$. For a $S \subseteq \Omega$ we write $\mathrm{dual}_A(S) := \{\mathrm{dual}_A(x) \mid x \in \Omega\}$.

Lemma 4.7. *The map* $\Omega \to \mathrm{dual}_A(\Omega)$ *given by* $x \mapsto \mathrm{dual}_A(x)$ *is a bijection and induces a permutation group isomorphism between the action of* G *on* Ω *and on* $\mathrm{dual}_A(\Omega)$.

Note that the set of t-transversals satisfy the requirements of A, with $m = rt$ and $\Omega = rB$. Even though H and G are isomorphic as groups in Definition 4.4, they are generally not as permutation groups. This is where the dual points come in. Given an anonymous set of points corresponding to the t-transversals, we will construct the dual points of those in rB, yielding an action of G on the dual points isomorphic to the non-existing points in rB.

Lemma 4.8 (Babai, Luks, Seress)**.** *If* $k < 2rt^2$*, then* $|G|$ *is quasi-polynomially bounded.*

Proof. Note that $n = \binom{k}{t}^r \geq (k/t)^{rt} \geq 2^{rt}$, so

$$|G| \leq |\mathfrak{S}_k \wr \mathfrak{S}_r| = (k!)^r r! < (2rt^2)^{2r^2t^2} r! < 2\log_2(n)^{4\log_2(n)^2}(\log_2(n)!),$$

which is quasi-polynomially bounded by Remark 2.2 and Lemma 2.3. □

It is therefore sufficient to consider the case where $k \geq 2rt^2$.

4.1 Cameron Schemes

First we will prove some statements about Cameron schemes $\mathcal{C}$ and the Cameron groups G acting on it. We thus assume we already have an identification between the domain of G and the set of transversals. These lemmata will then be used in the algorithm to recover such identification when it is not known a priori.

We call the orbits of the $G \subseteq \mathfrak{S}(A)$-action on A^2 the *orbitals* of G. As it turns out some relations in the Cameron scheme turn up as orbitals of G. Let $\Phi \subseteq A^2$ be the relation of the Cameron scheme corresponding to intersection pattern $(0, \ldots, 0)$ and Σ_i that of $(t-i, t, t, \ldots, t)$. For $\Delta \subseteq A^2$ let $\Delta(X)$ denote the set $\{Y \mid (X, Y) \in \Delta\}$.

Lemma 4.9 (Babai, Luks, Seress)**.** *The* Σ_i *and* Φ *are orbitals of* G.

Proof. First note that G preserves intersection patterns, so the orbitals are a refinement of the relations. Conversely, any pair of orbitals with given intersection *numbers* can be reached by G action from any other such pair, since the alternating group is t-transitive.

However, we have to take into account all permutations of the blocks and therefore the intersection numbers as well, since they give rise to the same intersection pattern. For Φ we are already done, since all intersection numbers are equal. For Σ_i, we note that there is just one intersection number we have to be able to move around, namely $t - i$. Here the transitivity of G acting on the blocks is sufficient. □

Using the argument in the above proof it is not difficult to find a relation of $\mathcal{C}$ that is not an orbital of G. Continuing Example 4.3, if the action of G on the blocks is the cyclic group C_3, then we cannot change $(X, Y) \in A^2$ with intersection numbers $(0, 10, 2)$ into a pair with intersection numbers $(0, 2, 10)$, so the relation with the corresponding intersection pattern is the disjoint union of two orbitals.

Lemma 4.10 (Babai, Luks, Seress)**.** *If $k \geq 2rt^2$, the largest and second smallest orbital of G are Φ and Σ_1 respectively.*

Proof. Note that Σ_0 is the smallest orbital, with $|\Sigma_0(X)| = 1$. For any $i > 1$,

$$|\Sigma_i(X)| = r\binom{t}{t-i}\binom{k-t}{i} > rt(k-t) = |\Sigma_1(X)|.$$

For any orbital Θ with intersection pattern $i, j, \ldots$ and $i \leq j < t$ we have

$$|\Theta(X)| \geq \binom{t}{i}\binom{k-t}{t-i}\binom{t}{j}\binom{k-t}{t-j} \geq t^2(k-t)^2 > rt(k-t) = |\Sigma_1(X)|$$

by just counting the images of (X, Y) under G in the blocks with intersection numbers i and j. We conclude that Σ_1 is the second smallest orbital. The proof for Φ is analogous. □

Lemma 4.11 (Babai, Luks, Seress)**.** *Let $(X, Y) \in \Sigma_1$, $P = \Phi(Y) \setminus \Phi(X)$ and $Q = \Omega \setminus \bigcup_{R \in P} \Phi(R)$. If $k \geq 2rt^2$, then $Q = \mathrm{dual}_A(x)$ where $\{x\} = X \setminus Y$.*

Proof. Since $(X, Y) \in \Sigma_1$, X and Y are equal in all but one of the copies of B, and share $t-1$ points in some other B. Call this block B_1, and let $\{x\} = X \setminus Y$. Note that $R \in P$ if and only if $x \in R$ and $Y \cap R = \emptyset$, since the only way for R to be disjoint from Y but not from X is to contain x. When $r = t = 1$, the statement is trivial, so assume $rt \geq 2$.

Let $Z \not\ni x$ be a transversal. Since $k \geq 2rt^2 \geq 4t$, we can then find a transversal R containing x that is disjoint from both Z and Y. It follows that $R \in P$, so $Z \in \bigcup_{R \in P} \Phi(R)$. Furthermore, if $Z \in \Phi(R)$ for some $R \in P$, then R cannot contain x. We conclude that $\bigcup_{R \in P} \Phi(R)$ is the set of transversals not containing x, which proves the claim. □

Now that we can recover rB as a whole, we finish off with identifying the individual B_i.

Lemma 4.12 (†)**.** *Let $x \in rB$ be given and let E be the set of dual points that intersect minimally with the dual point of x. Then $E \cup \{\mathrm{dual}(x)\} = \mathrm{dual}(B_i)$ for some i.*

Proof. Let $y \in rB$ be given and let x', y' be the duals of x, y respectively. If x and y are in the same block, then $|x' \cap y'|$ is a maximal intersection when $x = y$. If they are not equal, $|x' \cap y'| = \binom{k-2}{t-2} \cdot \binom{k}{t}^{r-1}$ if $t \geq 2$ and 0 otherwise. If x, y

are in different blocks, then $r \geq 2$ and $|x' \cap y'| = \binom{k-1}{t-1}^2 \cdot \binom{k}{t}^{r-2}$. Since $t < k$, we have

$$\frac{\binom{k-2}{t-2} \cdot \binom{k}{t}^{r-1}}{\binom{k-1}{t-1}^2 \cdot \binom{k}{t}^{r-2}} = \frac{k(t-1)}{t(k-1)} < 1,$$

so the dual points minimally intersecting x' are the others in the same block. It follows that $E \cup \{x'\}$ is the dual of some B. □

With this we can present the entire algorithm.

4.2 Retrieving the Structure

Here we assume again that the identification of Ω with some set of transversals A is unknown, as is the case in the main algorithm, and that $|G|$ is sufficiently large. Instead of using relations of an unknown Cameron scheme to find our dual points, we use the orbitals instead. From the above lemmata, the algorithm follows trivially.

Algorithm 4.13 (Babai, Luks, Seress)**.** Retrieve Cameron structure of G.

input : Cameron group $G \subseteq \mathfrak{S}(\Omega)$.
output: The set Γ of dual points of a block of the Cameron group.

1 Calculate the orbitals and identify Φ and Σ_1 by size (Lemma 4.10);
2 **for** $(X, Y) \in \Sigma_1$ **do**
3 $\quad P(X, Y) \leftarrow \{W \in \Omega \mid (Y, W) \in \Phi \wedge (X, W) \notin \Phi\}$;
4 $\quad Q(X, Y) \leftarrow \Omega \setminus \bigcup_{Z \in P(X,Y)} \{W \in \Omega \mid (Z, W) \in \Phi\}$;
5 $D \leftarrow \{Q(X, Y) \mid (X, Y) \in \Sigma_1\}$;
6 Pick any $d \in D$;
7 **for** $e \in D$ **do** $E(|e \cap d|) \leftarrow E(|e \cap d|) \cup \{e\}$;
8 **return** $E(k) \cup \{d\}$ *with* k *minimal such that* $E(k) \neq \emptyset$;

Theorem 4.14. *Algorithm 4.13 terminates in polynomial time and returns the correct result.*

Proof. Correctness follows from lemmata in the previous section and complexity is obvious. □

The algorithm returns only the dual points Γ of one block B_1. We can then find the other blocks by simply constructing the transitive closure of G acting on $\{B_1\}$, since G is transitive. We make a final remark on the Cameron group.

Lemma 4.15. *The set-wise stabiliser H of B_1 in G has $[G : H] \leq r \leq \log_2 n$, where we consider G, H as permutation groups on C.*

Proof. First note that $n = \binom{k}{t}^r \geq 2^r$, so $r \leq \log_2(n)$. Then

$$\begin{aligned}[G : H] &= [G : G \cap \mathfrak{S}(B_i) \wr_i \mathfrak{S}_{r-1}] \\ &\leq [\mathfrak{S}(B_i) \wr_i \mathfrak{S}_r : \mathfrak{S}(B_i) \wr_i \mathfrak{S}_{r-1}] = \frac{(k!)^r r!}{(k!)^r (r-1)!} = r.\end{aligned}$$

□

This completes our quest of finding the Cameron scheme.

Intermezzo

In this intermezzo we review the general outline of Babai's algorithm. Provided for the reader is a small diagram of the parts of the algorithm we treat in this thesis, to be found in Appendix A. It is advised to read this intermezzo with the diagram on hand.

We enter the algorithm through the incoming arrow at the top, and exit it through the outgoing dotted arrow when $G \subseteq \mathrm{Aut}(\mathfrak{x})$.

In the top-left corner of the diagram we have Luks' algorithm from Section 2.4. We find a block-system $\mathcal{P}$ of G which we stabilise, in Babai's algorithm only when the action H of G on $\mathcal{P}$ is sufficiently small. We apply weak reduction to the stabiliser and Ω splits into orbits. This is then followed by the Chain rule and a recursive call to the algorithm itself, collectively known as strong reduction.

If the action on the blocks is sufficiently large, Theorem 4.5 guarantees us that this must be a Cameron group. Using Algorithm 4.13, we then recover the bijection with transversals. Note that this bijection is with the blocks, since it is H that is the Cameron group, not (necessarily) G.

In Section 5 we will discuss the Going Down and Going Up arrows in the diagram. Using the Cameron scheme we obtain a group homomorphism $\phi : G \to \mathfrak{S}(\Gamma)$ with $\mathfrak{A}(\Gamma) \subseteq \mathrm{im}(G)$, which we call a giant representation. This allows us to forget the group G for a while and focus only on the strings, which have not received any attention up until now. Furthermore we construct a natural partitioning of Ω with respect to this ϕ, called the standard blocks. Collectively, we call this procedure 'going down'; we move our information in Ω, the strings, to Γ.

The algorithm considers the cases whether G is primitive or not separately. However, we won't consider this part of the algorithm in this thesis. In the first case, we can abuse the fact that our Cameron scheme on $\mathcal{P}$ becomes a Cameron scheme on Ω, since $\mathcal{P}$ is the singleton partitioning. This is the TopAction. In the second case, a lot more work is needed. In AggregateCertificates is where, according to himself, Babai's 'eureka moment' happened, and is probably the last part of the algorithm he completed.

In either case, we end up with a k-ary configuration (Section 6), from which we produce an embedded Johnson scheme $J : \Gamma \leftrightarrow \binom{M}{m}$, a special case of the Cameron scheme with only 1 block. That, or along the way we produced a coloured α-partition of Γ.

Then, we are back in Section 5, where we move the information we obtained in Γ back to Ω, which we call Going Up. This produces a reduced G and partitioning of Ω into orbits, to which we apply the chain rule again. However, one colour-class may be too large for this to become an effective reduction. For this colour-class we produce a smaller giant representation and enter the algorithm slightly differently.

5 Algorithmic Setup

In this section we treat the Going Down and Up part of the algorithm as discussed in the intermezzo. But first, we will need to obtain the giant representation.

Definition 5.1. For a general permutation group $G \subseteq \mathfrak{S}(\Omega)$, a homomorphism $\phi : G \to \mathfrak{S}(\Gamma)$ with $\mathfrak{A}(\Gamma) \subseteq \mathrm{im}(\phi)$ of which pull-backs can be computed in polynomial-time is called a *giant representation*.

The reason such morphisms are interesting, is because finding $\mathrm{Iso}_{\mathcal{C}}(A, B) \cap \phi(G)$ for objects A, B with underlying set Γ and category $\mathcal{C}$ is generally easy. For example, it is easy to describe all string isomorphisms $\mathrm{Iso}(\mathfrak{x}, \mathfrak{y})$, but not $\mathrm{Iso}(\mathfrak{x}, \mathfrak{y}) \cap G$. However, it is easy once we know $G = \mathfrak{S}(\Omega)$ or $G = \mathfrak{A}(\Omega)$. This giant representation allows us, to a certain extent, to forget the group G. We will find canonical structures with respect to $\mathfrak{x}$ and $\mathfrak{y}$ on Γ, and compute their $\phi(G)$-isomorphisms. The pull-back of this coset then reduces G if we choose interesting structures to find.

We obtain the giant representation from the Cameron scheme as follows.

Proposition 5.2. *Let $G \subseteq \mathfrak{S}(\Omega)$ be a permutation group with block-system $\mathcal{P} = \{P_1, \ldots, P_p\}$ such that the natural action $I = \mathrm{im}(G \to \mathfrak{S}(\mathcal{P}))$ on the blocks is a Cameron group. We can then produce a giant representation at quasi-polynomial multiplicative cost.*

Proof. Let $c : \mathcal{P} \leftrightarrow \binom{B}{t}^r$ be the Cameron scheme, which we extend with the projection $\pi_1 : \Omega \to \mathcal{P}$ onto the blocks to $f : \Omega \to \binom{B}{t}^r$. Consider the action $\psi : G \to \mathfrak{S}(B_i) \wr_i \mathfrak{S}_r$ induced by f, and let $M = \mathrm{im}(\psi) \cap \mathfrak{S}(B_1) \times \cdots \times \mathfrak{S}(B_r)$. Note that M is normal in $\mathrm{im}(\psi)$, and that by extension $N = \psi^{-1}(M)$ is normal in G. Furthermore, Lemma 4.15 tells us that $[G : N] \leq \log_2(p)!$, which is quasi-polynomially bounded in n. We apply Weak Luks Reduction to N, which incurs the mentioned multiplicative cost and makes N the group for which we need to give a giant representation. Let $J = \mathrm{im}(\pi_1 : N \to \mathfrak{S}(\mathcal{P}))$ be the natural action. Now, consider the composition ϕ of the maps

$$N \overset{\pi_1}{\twoheadrightarrow} J \overset{\varphi}{\hookrightarrow} \bigtimes_{i \leq r} \mathfrak{S}(B_i) \overset{\pi_2}{\to} \mathfrak{S}(B_1)$$

with φ the morphism induced by the Cameron scheme, and π_2 the restriction to the first block. We have by definition of the Cameron scheme that $\mathfrak{A}(B_1) \subseteq \phi(N)$. Furthermore, we can compute pull-backs of ϕ in quasi-polynomial time, since we can for π_1 by Lemma 2.21, for φ by Lemma 2.20, and for π_2 by Lemma 2.22. Hence ϕ is the claimed giant representation. □

Now that we are in the possession of a giant representation $\phi : G \to \mathfrak{S}(\Gamma)$, we continue by moving our information of string $\mathfrak{x}$ and $\mathfrak{y}$ through this ϕ.

5.1 Going Down

In this subsection we move the information in Ω, the strings $\mathfrak{x}, \mathfrak{y}$, to Γ.

For arbitrary giant representation $\phi : G \to \mathfrak{S}(\Gamma)$ with $G \subseteq \mathfrak{S}(\Omega)$, Babai defines the following.

Definition 5.3. Assume $|\Gamma| > 4 \log_2 |\Omega|$. For $x \in \Omega$, let $T(x) \subseteq \Gamma$ be the unique subset such that $|T(x)| < |\Gamma|/4$ and

$$\mathrm{Pstab}_{T(x)}(\mathfrak{A}(\Gamma)) \subseteq \phi(\mathrm{Pstab}_{\{x\}}(G)) \subseteq \mathrm{Sstab}_{T(x)}(\mathfrak{S}(\Gamma)).$$

Then the classes Ξ of the equivalence relation $\sim \subseteq \Omega^2$, where $x \sim y \Leftrightarrow T(x) = T(y)$, are the *standard blocks of G in Ω*.

It is not obvious that such $T(x)$ exist, nor that they are unique, and we will not prove the fact. Babai presents it in his Main Structure Theorem. The proof depends on a theorem of Jordan, which is treated in a book by Dixon and Mortimer [9].

However, in our case the standard blocks of ϕ as constructed in Proposition 5.2 are more easily characterised. For each $R \in \binom{B_1}{t}$ there is a standard block $S(R) \subseteq \Omega$ such that $x \in S(R)$ if and only if x when interpreted as a transversal has $x \cap B_1 = R$. These $S(R)$ are, with the work done in Section 4, certainly computable in polynomial time. Even though standard blocks are not defined when $|B_1| \not> 4 \log_2 |\Omega|$, we can always find the $S(R)$.

Lemma 5.4 (†)**.** *Assuming $|B_1| > 4 \log_2 |\Omega|$ and the soundness of Definition 5.3, the standard blocks of ϕ are the $S(R)$ as claimed.*

Proof. For $R \in \binom{B_1}{t}$, take any $x \in S(R)$. Firstly $|\Omega| \geq \binom{k}{t} \geq \left(\frac{k}{t}\right)^t \geq 2^t$, so $|R| = t \leq \log_2 |\Omega| < |B_1|/4$. Note that stabilisation of x in G stabilises the block R of x set-wise in B_1. Therefore $\phi(\mathrm{Pstab}_{\{x\}}(G)) \subseteq \mathrm{Sstab}_R(\mathfrak{S}(B_1))$. Secondly, a point-wise stabilisation of R in B_1 will certainly contain the set-wise stabilisation of $S(R)$, which is $\mathrm{Pstab}_{\{x\}}(G)$. We conclude by uniqueness that $R = T(x)$, so $S(R)$ is the standard block of x. □

In the above proof, we find that $|T(x)| = t$, which does not depend on x. In general, if x, y are in the same orbit of G, then $|T(x)| = |T(y)|$, which is another consequence of the proof we did not present.

Now, using the standard blocks we obtain strings $\mathfrak{x}', \mathfrak{y}' : \binom{\Gamma}{t} \to \mathbb{N}_0^\Sigma$ as follows.

Definition 5.5. Given a giant representation $\phi : G \to \mathfrak{S}(\Gamma)$ with $G \subseteq \mathfrak{S}(\Omega)$ and a string $\mathfrak{u} : \Omega \to \Sigma$, we define $\mathfrak{u}' : \binom{\Gamma}{t} \to \mathbb{N}_0^\Sigma$ the *string induced by ϕ* as

$$R \mapsto (\#\{x \in S(R) \mid \mathfrak{u}(x) = c\})_{c \in \Sigma}.$$

This concludes the objective of this subsection.

5.2 Going Up

Babai considers separately the case whether G is primitive or not, or equivalently whether or not the block-system $\mathcal{P}$ consists of singletons. Although both will not be treated in depth, the eventual goal will be the same; We want to find (relative) canonically embedded structures on Γ, specifically either a coloured α-partition or an embedded Johnson scheme. In this section we will define these structures and we will see how to move this information on Γ to Ω.

Definition 5.6. Let $\alpha \in (0, 1]$ and $\mathcal{P}$ a coloured equipartition of $\Gamma = \square(\mathcal{P})$ (Section 3). Then $\mathcal{P}$ is a *coloured α-partition* if for all partitions $A \in P \in \mathcal{P}$ we have $|A| \leq \alpha|\Gamma|$ and $|A| = 1$ only if $|P| = 1$.

Remark 5.7. Note that given a coloured equipartition

$$\mathcal{P} = (\{P_{11}, \ldots, P_{1q_1}\}, \ldots, \{P_{p1}, \ldots, P_{pq_p}\}),$$

we have $\mathrm{Aut}(\mathcal{P}) = \bigtimes_{1 \leq i \leq p} \mathfrak{S}(P_{ij}) \wr_j \mathfrak{S}_{q_i}$, for which we can find generators trivially.

We define analogously:

Definition 5.8. A *colouring* of Γ is a tuple $(C_1, \ldots, C_c)$ such that $\Gamma = C_1 \sqcup \cdots \sqcup C_c$. The morphisms are bijections $f : \Gamma \to \Gamma'$ such that $f(C_i) = C'_i$ for any other colouring $(C'_1, \ldots, C'_c)$ of Γ'. We call this an *α-colouring* if $|C_i| \leq \alpha|\Gamma|$ for all i.

Note that we have an obvious forgetful functor from coloured equipartitions to colourings that forgets the partitioning.

Definition 5.9. Let Γ, M be finite sets and let $2 \leq m \leq |M|/2$. Then a bijection $J : \Gamma \leftrightarrow \binom{M}{m}$ is called a *Johnson scheme*, and it has underlying set $\square(J) = \Gamma$. Given another Johnson scheme $K : \Gamma' \leftrightarrow \binom{M'}{m}$, its morphisms are functions $f : \Gamma \to \Gamma'$ for which there exists a map $g : M \to M'$, such that the following diagram commutes.

$$\begin{array}{ccc} \Gamma & \xrightarrow{\quad f \quad} & \Gamma' \\ {\scriptstyle J}\uparrow\downarrow & & \uparrow\downarrow{\scriptstyle K} \\ \binom{M}{m} & \overset{\exists g}{\dashrightarrow} & \binom{M'}{m} \end{array}$$

Again we actually only consider bijective morphisms, with as consequence that g in the above is unique if it exists. Finding a Johnson scheme with underlying set Γ is too much to ask. We are happy with an embedded Johnson scheme.

Definition 5.10. A functor $F : \mathcal{C} \to \mathcal{D}$ is a *canonical embedding of $\mathcal{D}$ in $\mathcal{C}$* if $\square(F(X)) \subseteq \square(X)$ for all $X \in \mathrm{obj}(\mathcal{C})$ and $F(f) : F(X) \to F(Y)$ is the restriction of f to $\square(F(X))$ for all $f \in \mathrm{Hom}_{\mathcal{C}}(X, Y)$.

This simply means that we want a Johnson scheme on a (large enough) subset of Γ that behaves on morphisms over Γ.

Both these structures induce a colouring of Γ; The coloured equipartition quite trivially by ignoring the partitioning of each colour, and the embedded Johnson scheme on $\Gamma' \subseteq \Gamma$ colours Γ as $(\Gamma', \Gamma \setminus \Gamma')$. We can move these colourings from Γ to Ω through ϕ analogously to Definition 5.5.

Definition 5.11. Given a giant representation $\phi : G \to \mathfrak{S}(\Gamma)$ with $G \subseteq \mathfrak{S}(\Omega)$ and a colouring $C = (C_1, \ldots, C_m)$ of Γ, we define the *colouring of* Ω *induced by* ϕ to be such that the colour of $x \in S(R)$ is $(\#(R \cap C_r))_{1 \leq r \leq m}$.

Note that again this colouring is canonical, and that this colouring is actually a colouring of the standard blocks.

The following lemma will be used, though we consider the proof to be insufficiently interesting to present and refer to Babai's manuscript.

Lemma 5.12. *If C is an α-colouring, then the induced colouring of Ω is a $\max(2/3, \alpha)$-colouring. For an arbitrary (non-α) colouring $(C_1, \ldots, C_c)$ the only induced colour Δ that can exceed the $2/3$ bound is of the form $(0, \ldots, 0, t, 0, \ldots, 0)$. We then call C_i the colour associated to Δ, where i is the place of t.* □

Any $g \in G$ preserving the colouring on Γ preserves the induced colouring on Ω. More specifically, $\mathrm{Aut}_G(\mathfrak{x}) \subseteq \phi^{-1} \mathrm{Aut}_{\phi(G)}(X) =: I$ where X is our coloured α-partition or Johnson scheme. Informally, we could replace G by I when looking for $\mathrm{Aut}_G(\mathfrak{x})$, and since I has orbits $\Delta_1, \ldots, \Delta_d$ we can apply the Chain rule. We will make this more rigorous in the next subsection. However, this reduction is not necessarily a good reduction, because we need $|\Delta_i| \leq \beta|\Omega|$ for all i and some predetermined constant β for Corollary 2.5 to apply. If we take $\beta > 2/3$, there is at most one Δ that can violate this restriction on size.

Lemma 5.13. *If Δ is an induced colour-class such that $|\Delta| > \beta|\Omega|$, we can find a giant representation $I \to \mathfrak{S}(\Gamma')$ where $|\Gamma'| \leq (1/2)|\Gamma|$.*

Proof. By Lemma 5.12, Δ corresponds to $\binom{C_i}{t}$ for some i. Therefore the induced action $\psi : I \to \mathfrak{S}(C_i)$ is well defined. If $C_i \leftrightarrow \binom{M}{m}$ was the colour of a Johnson scheme J, then we get in turn a map $\psi_+ : I \to \mathfrak{S}(M)$, where $|M| \leq (1/2)|\Gamma|$ certainly holds. This is a giant representation since $\mathfrak{A}(C_i) \subseteq \phi(I)$ and thus $\mathrm{Aut}_{\phi(I)}(J)$ is the induced action of $\mathfrak{A}(M)$ on C_i. Otherwise, our canonical structure must have been a coloured α-partition, and C_i must have been non-trivially partitioned into $\mathcal{P} = \{P_1, \ldots, P_p\}$. In this case the induced action on the blocks $\psi_+ : I \to \mathfrak{S}(\mathcal{P})$ is a giant representation, and since each $|P_i| \geq 2$, we have $|\mathcal{P}| \leq (1/2)|C_i| \leq (1/2)|\Gamma|$. □

We conclude that either we can reduce Ω to small orbits, or we have a large orbits with a significantly reduced giant representation.

5.3 Alignment of Strings

Here we will see why finding canonically embedded coloured α-partitions or Johnson-schemes in quasi-polynomial time is sufficient to complete the algorithm.

We assume functions `StringIsomorphism` and `StrIsoFromGiant` are declared, which the following algorithm is a part of by recursive calls to one another. We assume that `StrIsoFromGiant` can split $G \subseteq \mathfrak{S}(\Omega)$ into equal sized orbits when Γ in $\phi : G \to \mathfrak{S}(\Gamma)$ is smaller than some poly-logarithmic function of $|\Omega|$. This makes reducing Γ an alternative strategy to obtaining a good reduction. Formally, we use the following lemma.

Lemma 5.14. *Let $T : \mathbb{R}^2_{>0} \to \mathbb{R}_{>0}$ be a function where $T(n, \_)$ and $T(\_, m)$ are weakly increasing for fixed n, m. Let $\alpha, \beta \in (0, 1)$ and let p, q be quasi-polynomial where $q : \mathbb{R}^2_{>0} \to \mathbb{R}_{>0}$. Assume $T(n, m) \leq q(n, m)T(n, \beta m)$ and $T(n, 1) \leq p(n)T(\alpha n, \alpha n)$ for all n, m. Then $T(n, n)$ is quasi-polynomially bounded.*

Proof. By Lemma 2.4 $T(n, \_)$ is quasi-polynomially bounded by some function r. Then $T(n, n) \leq r(n)T(n, 1) \leq r(n)p(n)T(\alpha n, \alpha n)$ for all n, so $T(n, n)$ is quasi-polynomially bounded. □

Take T to be the complexity of the algorithm expressed in $|\Omega|$ and $|\Gamma|$. Then by the lemma, besides splitting Ω into orbits of size $\leq \alpha|\Omega|$, it is also sufficient to reduce Γ by a constant factor β at quasi-polynomial multiplicative cost.

Here we reap the rewards from the previous subsection in the form of an algorithm that computes $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$ given canonical structures relative to choices made to obtain them.

Besides the input to the function itself the algorithm has access to other data not given as parameters, which it was passed by currying, capturing from scope when we consider the function as lambda abstraction, or similar methods.

Algorithm 5.15. Align

input : Relative canonical coloured equipartitions $S^{\mathfrak{u}}$ of Γ or embedded Johnson-schemes $S^{\mathfrak{u}} = ((\Gamma')^{\mathfrak{u}} \leftrightarrow \binom{M^{\mathfrak{u}}}{m})$ with $|(\Gamma')^{\mathfrak{u}}| \geq (2/3)|\Gamma^{\mathfrak{u}}|$ for both $\mathfrak{u} \in \{\mathfrak{x}, \mathfrak{y}\}$.

data : Strings $\mathfrak{x}, \mathfrak{y} : \Omega \to \Sigma$, a giant representation $\phi : G \to \mathfrak{S}(\Gamma)$ with $G \subseteq \mathfrak{S}(\Omega)$

output: $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$

1 **if** $S^{\mathfrak{x}} \ncong S^{\mathfrak{y}}$ **then return** $\emptyset$;
2 Find any $\overline{\sigma} \in \mathrm{Iso}_{\phi(G)}(S^{\mathfrak{x}}, S^{\mathfrak{y}})$; `/* Lem 5.7 */`
3 Find any pull-back $\sigma \in \phi^{-1}(\overline{\sigma})$;
4 $G \leftarrow \phi^{-1}(\mathrm{Aut}_{\phi(G)}(S^{\mathfrak{x}}))$;
5 Let $(\Delta_1, \ldots, \Delta_d)$ be the colouring of Ω induced by $S^{\mathfrak{x}}$;
6 **foreach** Δ_i in order of decreasing size **do** (Chain rule)
7 | **if** $|\Delta| \leq (2/3)|\Omega|$ **then**
8 | | $\sigma G \leftarrow \sigma\pi_\Delta^{-1}(\texttt{StringIsomorphism}(\pi_\Delta(G), \mathfrak{x}|_\Delta, \sigma^{-1}(\mathfrak{y})|_\Delta))$;
9 | **else**
10 | | Let C be the colour of $S^{\mathfrak{x}}$ associated to Δ ; `/* Lem 5.12 */`
11 | | **if** $S^{\mathfrak{x}}$ is a coloured equipartition **then**
12 | | | Let Γ_+ be the partitioning of C in $S^{\mathfrak{x}}$;
13 | | | Let $\phi_+ : \pi_\Delta(G) \to \mathfrak{S}(\Gamma_+)$ the induced action on the blocks;
14 | | **else** ($S^{\mathfrak{x}}$ is a Johnson scheme)
15 | | | Let $C \leftrightarrow \binom{\Gamma_+}{m_+}$ the Johnson scheme;
16 | | | Let $\phi_+ : \pi_\Delta(G) \to \mathfrak{S}(\Gamma_+)$ the induced giant representation;
17 | | $\sigma G \leftarrow \sigma\pi_\Delta^{-1}\texttt{StrIsoFromGiant}(\mathfrak{x}|_\Delta, \sigma^{-1}(\mathfrak{y})|_\Delta, \phi_+^{\mathfrak{x}}, \phi_+^{\mathfrak{y}})$;
18 **return** σG;

Lemma 5.16. *Algorithm 5.15 returns* $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$ *relative to choices made to obtain the structures, at quasi-polynomial multiplicative cost.*

Proof. First we note that it is easy to find $\phi(G)$ isomorphisms of coloured partitions or Johnson schemes, since $\phi(G) = \mathfrak{A}(\Gamma)$ or $\phi(G) = \mathfrak{S}(\Gamma)$, so line 1 and 2 are executed in polynomial time. If $S^{\mathfrak{x}}$ and $S^{\mathfrak{y}}$ are not isomorphic (1), then clearly $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) = \emptyset$ with respect to choices made to obtain $S^{\mathfrak{x}}, S^{\mathfrak{y}}$. Clearly every relative isomorphism $\mathfrak{x} \cong \mathfrak{y}$ is an isomorphism of $S^{\mathfrak{x}} \cong S^{\mathfrak{y}}$, so $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) \subseteq \sigma\phi^{-1}(\mathrm{Aut}(S^{\mathfrak{x}}))$, justifying mathematically the reductions on line 3 and 4. Since ϕ is a giant representations, lines 3 and 4 execute in polynomial time as well.

With the updated G (4), the $\Delta_i \subseteq \Omega$ are now G-stable. If each Δ_i is small enough (7), then we reduce string isomorphism on Ω to $d \leq n$ instances of string isomorphism on the Δ_i with $|\Delta_i| \leq (2/3)|\Omega|$. If there is a larger Δ (9), then there is a large colour-class in C in $S^{\mathfrak{x}}$ associated to Δ (10) by Lemma 5.12, which must then either be partitioned if $S^{\mathfrak{x}}$ is a coloured α-partition (11), or a Johnson scheme (14). In either case we obtain a giant representation when we consider the action of G on Γ_+ by Lemma 5.13. Again in either case we reduce Γ by a constant factor.

Here it is important we start with the large orbital. If we considered a small orbit first, we reduce G, after which ϕ might not be a giant representation anymore.

If `StringIsomorphism` takes multiplicative cost $p(|\Omega|)$ to arrive at `Align` and `StrIsoFromGiant` multiplicative cost $q(|\Omega|, |\Gamma|)$, then

$$T(n, m) \leq np(n)T(2n/3, 2n/3) + q(n, m)T(n, m/2) + a(n, m),$$

where T is the worst-case time complexity of `StringIsomorphism` taken over all problem instances with $|\Omega| \leq n$ and $|\Gamma| \leq m$. If p, q, a are quasi-polynomial, then so is T by Lemma 5.14. □

With this algorithm in place, the only thing we need to do now is provide the algorithm `StrIsoFromGiant` that finds the relative canonical structures at quasi-polynomial additive and multiplicative cost. This however, will mostly be left untreated in this thesis. The next section will introduce an important structure used in the Extended Design Lemma, which is a part of `StrIsoFromGiant`.

6 Configurations

A powerful tool used in the treatment of the graph isomorphism problem is the coherent configuration. From the refinement of a graph to a coherent configuration one can often confirm or reject isomorphism for random graphs in polynomial time [10]. An extension of this refinement procedure, called Individualisation/Refinement, works for all graphs, but was shown to require exponential time on some family of graphs [11]. Despite this, we can make use of configurations.

Definition 6.1. Let Ω be a set and let $k \in \mathbb{N}_1$. Then $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ is a *k-ary relational structure* if no R_i is empty and the R_i partition Ω^k. Each R_i is called a *k-ary relation* of $\mathfrak{X}$ and for each $\vec{x} \in R_i$ we call i the *colour* of x, written as $c(\vec{x}) = i$.

Definition 6.2. A k-ary relational structure $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ is called a *(k-ary) configuration* if

1. (Shape axiom) for some $(x_1, \ldots, x_k) \in R_m$ we have $x_i = x_j$, then for all $(y_1, \ldots, y_k) \in R_m$ we have $y_i = y_j$;
2. (Permutation axiom) for each $\sigma \in \mathfrak{S}_r$ and i we have $R_i^\sigma = R_j$ for some j.

Here $S^\sigma := \{(x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(k)}) \mid (x_1, \ldots, x_k) \in S\}$. We call $\mathfrak{X}$ *homogeneous* if the diagonal $\mathrm{diag}_k(\Omega) := \{(x, \ldots, x) \mid x \in \Omega\}$ has a unique colour.

Configurations come up as generalisation of various concepts. A simple graph $G = (V, E)$ for example can be interpreted as a homogeneous binary configuration $\mathfrak{X}_G = (V; \mathrm{diag}_2(V), E, V^2 \setminus (E \cup \mathrm{diag}_2(V)))$, where we have a colour for the diagonal, which is interpreted as the set of vertices, a colour for the edges, and a colour for everything else. For a group acting on Ω the orbits of the action on Ω^k form a k-ary configuration. The other way around, we can interpret a binary configuration as a coloured complete digraph, where the edges and vertices have different colours. For $k = 1$, a k-ary relational structure is just a coloured set. For larger k, the interpretation of $\vec{x} \in R_i$ becomes that of a coloured path in the complete graph on Ω.

Definition 6.3. A *weak isomorphism* between two k-ary configurations $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ and $\mathfrak{X}' = (\Omega'; R'_1, \ldots, R'_r)$ is a bijection $f : \Omega \leftrightarrow \Omega'$ such that for each i there exists a j such that $(f \times \ldots \times f)(R_i) = R'_j$. A *(strong) isomorphism* is a weak isomorphism that preserves colour, i.e. $i = j$.

Definition 6.4. A *refinement* of a k-ary relational structure is a refinement of the relations.

Remark 6.5. For strong isomorphism, the order of the relations is relevant, so when giving a refinement of the relations, an order for these is needed. Furthermore, when constructing a canonical refinement this must also be done independently of the specific elements the partition contains.

In this thesis we only consider strong isomorphisms, continuing the convention that isomorphisms preserve colours. For our purposes it is sufficient to consider the isomorphisms where the underlying sets are equal, and again bijections become permutations. Note that the strong isomorphisms on the configuration corresponding to a graph are exactly the graph isomorphisms.

In the case of graphs, each isomorphism preserves vertex degree. For configurations, an analogous statement holds for the for each relation when we extend the concept of degree. A refinement of the colours of the vertices such that they all have the same degree in a specific relation will therefore have the same automorphism group. This inspires the following definition:

Definition 6.6. Let $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ be a configuration. For any $i_1, \ldots, i_k \in \{1, \ldots, r\}$ and $\vec{x} \in \Omega^k$ we write

$$P_{\vec{x}}(i_1, \ldots, i_k) := \{y \in \Omega \mid (\forall j)\ c(\vec{x}^y_j) = i_j\},$$

where $\vec{x}^y_j$ is $\vec{x}$ with y substituted on the j-th place.

Remark 6.7. For each $\sigma \in \mathrm{Aut}(\mathfrak{X})$ we have that $\#P_{\vec{x}}(\vec{\imath}) = \#P_{\sigma(\vec{x})}(\vec{\imath})$.

Definition 6.8. We call a configuration *coherent* if $\#P_{\vec{x}}(i_1, \ldots, i_k)$ is equal for all $\vec{x}$ of the same colour i_0. We then write $p(i_0; i_1, \ldots, i_k) = \#P_{\vec{x}}(i_1, \ldots, i_k)$. These p are called the *structure constants* of $\mathfrak{X}$.

Proposition 6.9. *For each configuration $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ there exists a canonical refinement $\mathfrak{X}^*$ of $\mathfrak{X}$ that is a coherent and* $\mathrm{Aut}(\mathfrak{X}) = \mathrm{Aut}(\mathfrak{X}^*)$.

Proof. Let $n = |\Omega|$ and for each map $M : \{1, \ldots, r\}^k \to \{1, \ldots, n\}$ define

$$S(j, M) := \{\vec{x} \in R_j \mid (\forall \vec{\imath} \in \{1, \ldots, r\}^k)\ \#P_{\vec{x}}(\vec{\imath}) = M(\vec{\imath})\}.$$

The non-empty S form a refinement $\mathfrak{X}^*$ of $\mathfrak{X}$, so $\mathrm{Aut}(\mathfrak{X}^*) \subseteq \mathrm{Aut}(\mathfrak{X})$, and by Remark 6.7 $\mathrm{Aut}(\mathfrak{X}) \subseteq \mathrm{Aut}(\mathfrak{X}^*)$. Repeating this construction with $\mathfrak{X}^*$, $\mathfrak{X}^*$ must eventually stabilise, and this stable configuration is coherent. If in each step we order the j and M lexicographically, the $S(j, M)$ have a canonical order, so $\mathfrak{X}^*$ is a canonical refinement of $\mathfrak{X}$. □

The refinement obtained is called the *Weisfeiler-Leman refinement of* $\mathfrak{X}$ [12]. When k is a constant, the construction of $\mathfrak{X}^*$ as in Proposition 6.9 is polynomial

in n, as the number of iteration steps is bounded above by n^k; In each non-final step a new relation must be introduced, and each step is clearly polynomial if we only consider the functions M for which a $S(j, M)$ will be non-empty. A non-naïve approach to constructing this refinement is given by Immerman and Lander.

Theorem 6.10 (Immerman, Lander [13])**.** *Given a k-ary configuration $\mathfrak{X}$ over Ω, where $|\Omega| = n$, we can find $\mathfrak{X}^*$ in $O(k^2 n^{k+1} \log(n))$ time.*

Note that if $k \in O(\log(n))$, then the algorithm runs in quasi-polynomial time.

Remark 6.11. Sometimes when working with structure constants, we require that $\vec{x}_j^y$ can be in any relation. We can simply take $\bigsqcup_{a \leq r} P_{\vec{x}}(\vec{\imath}_j^a)$ and write a question mark for any such place, like $P_{\vec{x}}(i_1, \ldots, i_{k-1}, ?)$. Since the union is disjoint, we may define $p(i_0; i_1, \ldots, i_{k-1}, ?)$ analogously.

6.1 Coherent Configurations

In this subsection we will prove some properties of coherent configurations that are more substantial than the structure constants. This requires some definitions.

Definition 6.12. Given a relation R in a k-ary configuration, we define its *shape* to be the unordered partitioning of $\{1, \ldots, k\}$ such that i, j share a partition if and only if $\vec{x}_i = \vec{x}_j$ for all $\vec{x} \in R$. We call the colours with shape $\{1, \ldots, k\}$ the *primary colours*, those with shape $\{1\}, \ldots, \{k\}$ the *k-ary colours*.

Note that by the shape axiom, the shape of a relation in a configuration is well-defined.

Definition 6.13. Let $1 \leq s \leq k$ and let $\mathfrak{X} = (\Omega; R_1, \ldots, R_N)$ be a k-ary configuration. Let $R_i'' = \{(x_1, \ldots, x_{s-1}, y) \mid (x_1, \ldots, x_{s-1}, y, \ldots, y) \in R_i\}$. Then $\mathrm{skel}_s(\mathfrak{X}) := (\Omega; R_1', \ldots, R_M')$ is the *s-skeleton of $\mathfrak{X}$*, where $R_1', \ldots, R_M'$ is the subsequence of $R_1'', \ldots, R_N''$ of non-empty sets.

Here we define the skeleton by the elements such that the last couple of places are equal. However, if we chose a different set S of $k-s+1$ places that need to be equal, then by the permutation axiom the resulting configurations would differ only by a reordering of the colours, i.e. they are weakly isomorphic through the identity map. For this we use the notation $\mathrm{skel}(\mathfrak{X}, S)$.

Lemma 6.14. *If $\mathfrak{X}$ is coherent, then so is* $\mathrm{skel}(\mathfrak{X}, S)$.

Proof. Since each relation in the original configuration has a well defined shape, it is either not included in the skeleton at all, or in its entirety modulo some equal places. It follows that $\mathrm{skel}(\mathfrak{X}, S)$ is coherent when $\mathfrak{X}$ is, since we can just use the structure constants from $\mathfrak{X}$ in the skeleton. □

Definition 6.15. Let $C \subseteq \Omega$ be a subset and $\mathfrak{X} = (\Omega; R_1, \ldots, R_N)$ be a k-ary configuration. Then $\mathrm{ind}(\mathfrak{X}, C) := (C; R_1', \ldots, R_M')$ is the *induced configuration in C*, where the $R_1', \ldots, R_M'$ is the subsequence of $R_1 \cap C^k, \ldots, R_N \cap C^k$ of non-empty sets.

Lemma 6.16. *Assume $\mathfrak{X}$ is coherent. For $(x_1, \ldots, x_k) \in R_i$, the colour of $(x_1, \ldots, x_{k-2}, x_k, x_k)$ only depends on i.*

Proof. Let $\vec{x} \in R_i$ be given and define $u = x_k$ and $j = c(\vec{x}^u_{k-1})$. Note that $p(i; ?, \ldots, ?, j, i) \geq 1$, since $u \in P_{\vec{x}}(?, \ldots, ?, j, i)$. Then by coherence, for any $\vec{y} \in R_i$, there exists a $v \in P_{\vec{y}}(?, \ldots, ?, j, i)$. Since $\vec{y}^v_{k-1} \in R_j$, it follows from its shape that $v = y_k$, so $c((x_1, \ldots, x_{k-2}, x_k, x_k)) = c((y_1, \ldots, y_{k-1}, y_k, y_k))$ for any two $\vec{x}, \vec{y} \in R_i$. □

Corollary 6.17. *For $(x_1, \ldots, x_k) \in R_i$, the colour of $(x_j, \ldots, x_j)$ only depends on i and j.*

Proof. By the permutation axiom, we can assume without loss of generality that $j = k$. When $k = 1$ the statement is trivial. Otherwise, we use Lemma 6.16 and reduce to the $(k-1)$-skeleton and apply induction. □

It follows that all colours have an associated sequence of primary colours.

Corollary 6.18. *If $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ is coherent and $C \subseteq \Omega$ is a union of vertex colour-classes, then* $\mathrm{ind}(\mathfrak{X}, C)$ *is coherent.*

Proof. As for the skeleton, each relation is included entirely or not at all. □

Note that the construction of the skeleton is canonical, and the same holds for the induced configuration if we choose C canonically.

Definition 6.19. Let $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ be a k-ary configuration. Then $d_m(v, i) := |\{\vec{x} \in R_i \mid x_m = y\}|$ is called the *m-th visit degree of v in R_i*.

Lemma 6.20. *Let $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ be a coherent k-ary configuration. For $v \in \Omega$ the size of $d_m(v, i)$ depends only on i, m and $c(v)$.*

Proof. We assume without loss of generality that $m = k$. For $k = 1$ the statement is trivial. Now assume $k > 1$ and that the statement holds for $k - 1$. Let $j = c(x_1, \ldots, x_{k-2}, x_k, x_k)$ for all $\vec{x} \in R_i$, which depends only on i and m. Then the size of $I(v) = \{(x_1, \ldots, x_{k-2}, v, v) \in R_j\}$ only depends on i, m and $c(v)$ when we apply the induction hypothesis to the $(k-1)$-skeleton. Then

$$|d_m(v, i)| = \sum_{\vec{x} \in I(v)} |P_{\vec{x}}(?, \ldots, ?, i, ?)| = |I(v)| \cdot p(j; ?, \ldots, ?, i, ?)$$

depends only on i, m and $c(v)$. □

Definition 6.21. A coarsest configuration on Ω^k is called a *clique*.

Lemma 6.22. *$\mathfrak{X}$ is a clique if and only if* $\mathrm{Aut}(\mathfrak{X}) = \mathfrak{S}(\Omega)$ *if and only if for each shape there is a unique relation in $\mathfrak{X}$ with that shape.*

Proof. Easy. □

The following proposition is used by Babai in his proof on the correctness of the Design Lemma, and he calls it twin awareness. However, he only presents a proof for the case $k = 2$. We extend it to hold in the general case.

Proposition 6.23 (†)**.** *Let $\mathfrak{X} = (\Omega; R_1, \dots, R_r)$ be a k-ary configuration and let $x, y \in \Omega$ with $e = c(x, \dots, x, y)$. If $(x\ y) \in \mathrm{Aut}(\mathfrak{X})$, then $(u\ v) \in \mathrm{Aut}(\mathfrak{X})$ for all $(u, \dots, u, v) \in R_e$.*

Proof. When $\mathfrak{X}$ is unary, then the statement is trivial. In the case $\mathfrak{X}$ is binary, the statement is proven by Babai. We will prove the statement for larger k by induction, so let $k > 2$ be given and assume the statement holds for $k - 1$.

Assume $(x\ y) \in \mathrm{Aut}(\mathfrak{X})$ with $(x, \dots, x, y) \in R_e$, and let $(u, \dots, u, v) \in R_e$ be given. We need to prove that $(u\ v)$ preserves the colour of each $\vec{a} = (a_1, \dots, a_k) \in \Omega^k$.

If $\vec{a} \in R_i$ is not of k-ary colour, we assume without loss of generality that $a_{k-1} = a_k$. We reduce to the $(k-1)$-skeleton, where the statement holds by assumption. There $c((u\ v)(a_1, \dots, a_{k-1})) = c((a_1, \dots, a_{k-1}))$, hence $(u\ v)(a_1, \dots, a_k) \in R_i$. We conclude that $(u\ v)$ respects the non-k-ary colours.

We now consider a relation R_i with k-ary colour. For any $\vec{b} = (b_1, \dots, b_k) \in R_i$ with $u, v \notin \{b_1, \dots, b_k\}$, we certainly have $(u\ v)\vec{b} = \vec{b} \in R_i$. Therefore assume without loss of generality $b_k = u$.

Let R_l be the relation corresponding to R_i where the $(k-3)$-th place equals the $(k-2)$-th, which only depends on i by Lemma 6.16. Write

$$\begin{aligned}
J_{u,v} &= \{(a_1, \dots, a_{k-1}) \mid c(a_1, \dots, a_{k-1}, u) = i\},\\
K_{u,v} &= \{(a_1, \dots, a_{k-1}) \mid c(a_1, \dots, a_{k-1}, u) = c(a_1, \dots, a_{k-1}, v) = i\},\\
L_{u,v} &= \{(a_1, \dots, a_{k-1}) \mid c(a_1, \dots, a_{k-1}, u) = i \wedge v \in \{a_1, \dots, a_{k-1}\}\},\\
M_{u,v} &= \{(a_1, \dots, a_{k-3}) \mid c(a_1, \dots, a_{k-3}, u, u, v) = l\},\\
N_{u,v} &= \{(a_1, \dots, a_{k-2}) \mid c(a_1, \dots, a_{k-2}, u, v) = i\}.
\end{aligned}$$

Note that $|J_{u,v}| = m_k(u, j)$ only depends on the colours of u by Lemma 6.20. For each $(a_1, \dots, a_{k-3}, u, u, v) \in R_l$ we have a $(u\ x)(v\ y)(a_1, \dots, a_{k-3}, u, u, v) \in R_l$ and vice versa by the non-k-ary case, so $|M_{u,v}|$ only depends on the colour of $(u, \dots, u, v)$. Then

$$\begin{aligned}
|N_{u,v}| &= \sum_{(a_1,\dots,a_{k-3}) \in M_{u,v}} \left|P_{(a_1,\dots,a_{k-3},u,u,v)}(?, \dots, ?, i, ?, ?)\right|\\
&= |M_{u,v}| \cdot p(l; ?, \dots, ?, i, ?, ?).
\end{aligned}$$

and $|L_{u,v}| = (k-1)|N_{u,v}|$ both only depend on the colour of $(u, \dots, u, v)$. Now let j be such that $R_i^{(k-1\ k)} = R_j$. Lastly

$$\begin{aligned}
|K_{u,v}| &= \sum_{(a_1,\dots,a_{k-2}) \in N_{u,v}} \left|P_{(a_1,\dots,a_{k-2},u,v)}(?, \dots, ?, i, j)\right|\\
&= |N_{u,v}| \cdot p(i; ?, \dots, ?, i, j)
\end{aligned}$$

only depends on the colour of $(u, \dots, u, v)$.

Note that $K_{u,v} \sqcup L_{u,v} \subseteq J_{u,v}$. By definition of these sets, the equality holds for $K_{x,y} \sqcup L_{x,y} = J_{x,y}$. Therefore $|K_{x,y}| + |L_{x,y}| = |J_{x,y}|$. But since these sizes

only depend on the colour of $(x, \dots, x, y)$, we get $|K_{u,v}| + |L_{u,v}| = |J_{u,v}|$ as well. Hence the inclusion we had must be an equality, so $K_{u,v} \sqcup L_{u,v} = J_{u,v}$.

Since $\vec{b} \in J_{u,v}$, either $\vec{b} \in K_{u,v}$, in which case $(u\ v)\vec{b} = (b_1, \dots, b_{k-1}, v) \in R_i$, or $\vec{b} \in L_{u,v}$. In this case there also exists an $\vec{a} = (a_1, \dots, a_{k-1}, x) \in R_i$ with y on the same place p as v in $\vec{b}$, because the size of $N_{u,v}$ only depends on the colour of $(u, \dots, u, v)$. But then $(x\ y)\vec{a} \in R_i$, so $R_i^{(p\ k)} = R_i$, and thus $(u\ v)\vec{a} \in R_i$. In all cases, $(u\ v)$ respects the colouring, so $(u\ v) \in \mathrm{Aut}(\mathfrak{X})$. □

6.2 Binary Configurations and UPCCs

In this subsection we consider the binary subspecies of configurations and their relations to the other categories we consider.

Lemma 6.24. *Let $\mathfrak{X} = (\Omega; R_1, \dots, R_r)$ be a binary coherent configuration and $i \leq r$. Let $A, B \subseteq \Omega$ be the colour-classes of the first and second place of R_i respectively. If $A = B$, then (A, R_i) is a regular graph. Otherwise, $(A, B; R_i)$ is a semi-regular bipartite graph.*

Proof. This is a direct consequence of Lemma 6.20. □

In either of the above cases, we refer to such graphs as the *i-th constituent digraph.*

Definition 6.25. Let $\mathfrak{X} = (\Omega; R_1, \dots, R_r)$ be a homogeneous binary coherent configuration. Then $\mathfrak{X}$ is called *primitive* if (Ω, R_i) is a strongly connected digraph non-diagonal colours i. Additionally, $\mathfrak{X}$ is called *uniprimitive (or $\mathfrak{X}$ is a UPCC)* if it is primitive and not a clique.

Remark 6.26. Let $\mathfrak{X}$ be an imprimitive binary configuration. If it is non-homogeneous, $\mathrm{skel}_1(\mathfrak{X})$ is a canonical non-trivial colouring of Ω. If there exists a secondary colour with smallest index i such that the graph (Ω, R_i) is not strongly connected, we find a canonical partitioning of Ω into the connected components $\{B_1, \dots, B_m\}$ of this graph. Combining the two, $\mathfrak{X}$ yields a non-trivial canonical coloured equipartition of Ω.

Conversely, if we have configuration $\mathfrak{X}$ and a canonical coloured equipartition E, we may refine $\mathfrak{X}$ with E by splitting the vertex colours by the colours of E, and the edge colours by whether the pair of vertices are in the same partition or not. If E is non-trivial, the resulting configuration is imprimitive.

Definition 6.27. A k-regular graph $G = (V, E)$ is called (k, λ, μ)*-strongly regular* if every two vertices have λ common neighbours if they are adjacent, and μ if they are not.

Lemma 6.28 (†)**.** *Let $\mathfrak{X} = (\Omega; R_1, R_2, R_3)$ be a homogeneous binary coherent configuration with primary colour 1 and secondary colours 2 and 3. If $R_2^{(1\ 2)} = R_2$, then $G = (\Omega, R_2)$ is a strongly regular graph.*

Proof. By Lemma 6.24, G is regular. Note that the shared neighbours of $u, v \in \Omega$ are $P_{(u,v)}(2, 2)$, using the fact that $R_2^{(1\ 2)} = R_2$. These numbers are $\lambda = p(2; 2, 2)$ if u, v are adjacent, and $\mu = p(3; 2, 2)$ if they are not, so G is (k, λ, μ)-strongly regular. □

Lemma 6.29 (†)**.** *Let $G = (V, E)$ be a (k, λ, μ)-strongly regular graph. Then the configuration $\mathfrak{X}_G$ induced by G is coherent.*

Proof. We clearly have for $i, j \in \{1, 2\}$ that

$$p(1; i, j) = \begin{pmatrix} 1 & 0 \\ 0 & k \end{pmatrix}_{i,j}, \quad p(2; i, j) = \begin{pmatrix} 0 & 1 \\ 1 & \lambda \end{pmatrix}_{i,j}, \quad p(3; i, j) = \begin{pmatrix} 0 & 0 \\ 0 & \mu \end{pmatrix}_{i,j},$$

by similar arguments as in Lemma 6.28. Note that $P_{(u,v)}(?, j)$ is the neighbourhood of v in R_j, which has sizes 1, k and $n - k - 1$ for colours 1, 2 and 3 respectively. But these are also the row sums and column sums of the 3×3-matrices $(p(h; i, j))_{i,j}$ for all h. With this information, we can compute all structure constants, and the configuration is coherent. □

We see that coherent configurations can also be interpreted as a generalisation of strongly regular graphs. With this correspondence, we can supply some counterexamples to some common questions.

Example 6.30. The Chang graphs are three $(12, 6, 4)$-strongly regular graphs on 28 vertices [14]. These graphs have a property rare among strongly-regular graphs, namely that they have a non-trivial automorphism group [15]. Even more striking is that it is not even transitive, as it has two orbits; one of size 4 and one of 24. Since the orbits have different sizes, they can be canonically ordered to form a non-trivial colouring C of V. By Lemma 6.29, $\mathfrak{X}_G$ is both coherent and homogeneous. However, we can refine $\mathfrak{X}_G$ with C and obtain a coherent configuration $\mathfrak{X}'$ with $\mathrm{Aut}(\mathfrak{X}_G) = \mathrm{Aut}(\mathfrak{X}')$. We conclude that for any configuration $\mathfrak{X}$, the refinement $\mathfrak{X}^*$ is not necessarily the finest coherent configuration with the same automorphism group.

Furthermore, the Chang graphs and their complements are connected and non-trivial, meaning that $\mathfrak{X}_G$ is a UPCC. Since $\mathrm{Aut}(\mathfrak{X}_G)$ is non-transitive, it is certainly neither primitive nor uniprimitive, even though the naming might suggest otherwise.

7 Concluding Remarks

In this section we make a small number of remarks about proofs in Babai's manuscript that we were not able to treat in this thesis. This is not intended for readers of this thesis alone, but for those who have read or are also reading Babai's work. Mostly because some definitions will be used that have not been treated in this thesis so far.

Design Lemma

Babai makes several claims, that the coherent configuration is 'aware' of certain properties of elements in Ω. He does not prove them, which is justified by the fact that we can extend the algorithm in the Design Lemma slightly to force this 'awareness' either way, while maintaining polynomial additive cost. A proof of these claims would be interesting regardless. For one we only need a small extension of Babai's 'twin awareness', which we give in Proposition 6.23.

Corollary 7.1 (†)**.** *Let $\mathfrak{X}$ be a coherent configuration with* $\mathrm{SymDef}(\mathfrak{X}) \geq 1-\alpha$ *and vertex colour-class C of size $|C| \geq \alpha|\Omega|$. If* $\mathrm{skel}_2(\mathrm{ind}(\mathfrak{X}, C))$ *is a clique, then $(x\ y) \notin \mathrm{Aut}(\mathfrak{X})$ for all different $x, y \in C$.*

Proof. Assume for such x, y that $(x\ y) \in \mathrm{Aut}(\mathfrak{X})$. Since $\mathrm{skel}_2(\mathrm{ind}(\mathfrak{X}, C))$ is a clique, we have $c(x, \ldots, x, y) = c(u, \ldots, u, v)$ for all $u, v \in C$. From Proposition 6.23 it follows that $\mathfrak{S}(C) \subseteq \mathrm{Aut}(\mathfrak{X})$. However, $\mathrm{SymDef}(\mathfrak{X}) \geq 1-\alpha$ and $|C| \geq \alpha|\Omega|$, a contradiction. □

UPCC-to-Bipartite

We are presented with a UPCC $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$, where we assume R_1 is the diagonal, and a point $x \in \Omega$. In the proof we are required to find two relations such that the following holds.

Lemma 7.2 (†)**.** *Let $\mathfrak{X} = (\Omega; R_1, \ldots, R_r)$ be a UPCC and let $x \in \Omega$ and $3 \leq j \leq r$ be given. Then there exists an $i \geq 3$, canonical relative to x, such that $(x, z) \in R_2$, $(x, y) \in R_i$ and $(z, y) \in R_j$ for some $y, z \in \Omega$.*

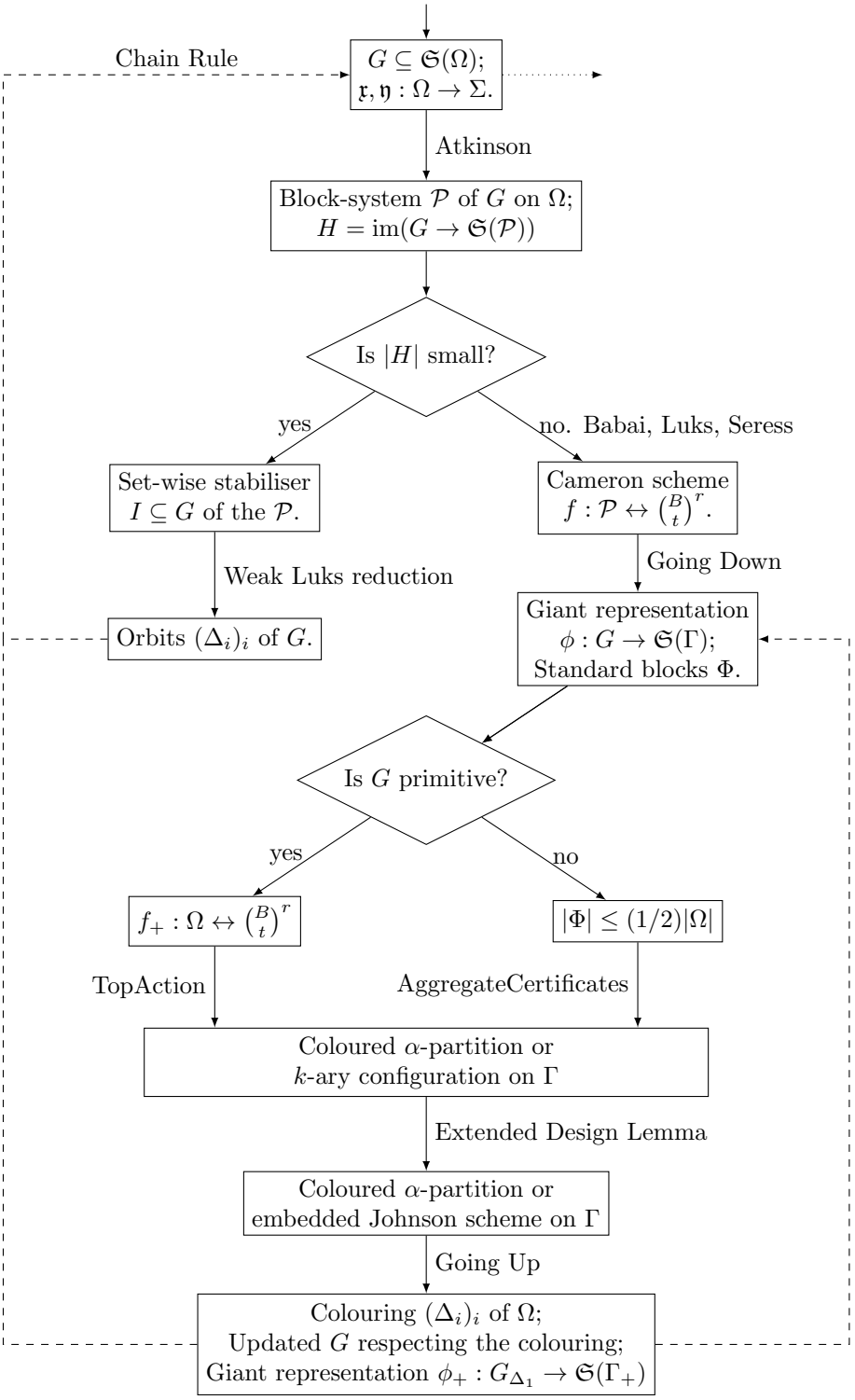
Proof. Since R_2 is not empty and $\mathfrak{X}$ is homogeneous, there exists a $w \in \Omega$ such that $(x, w) \in R_2$. Because $\mathfrak{X}$ is not a clique, R_j is a relation of $\mathfrak{X}$, and since $\mathfrak{X}$ is primitive R_j is connected. Let then $w = a_0 \to a_1 \to \ldots \to a_m = x$ be a path in R_j. Because $(x, w) \notin R_j$, we have that $m \geq 2$. Let l be maximal such that $(x, a_l) \in R_2$ and note that $l \leq m-2$. Let $z = a_l$ and $y = a_{l+1}$, then $(x, z) \in R_2$, $(z, y) \in R_j$ and $(x, y) \in R_i \neq R_2$ by the maximality of l. Now that such y, z exist, we can take i to be minimal such that these y, z exist, and such a choice is then canonical. □

Reduce-Part2-by-Color

Given is a bipartite graph $B = (V_1, V_2; E)$ such that $|V_2| \leq \alpha|V_1|$ and there are no twins in V_1, and a colouring (C_1, C_2) of V_1. From this the procedure `Reduce-Part2-by-Color` returns $B_+ = (C_i, V_2; E \cap C_i \times V_2)$ for some $i \in \{1, 2\}$ such that $\mathrm{SymDef}_{B_+}(V_1) \leq \alpha$.

At many points in the Extended Design Lemma, we will consider the case where B is biregular. However, when we call `Reduce-Part2-by-Color`, the resulting graph generally not biregular anymore. We can only guarantee it when C_1, C_2 are colour-classes of the coherent configuration induced by B (Lemma 6.18). This is not a problem when we intend to call `Bipartite Split-or-Johnson` with this graph, since it takes general bipartite graphs as arguments. However, at subcase 2 of the Block Design case and subcase (II) of the UPCC case, we require biregularity. The author of this thesis was not able to prove the correctness of these steps, nor does he propose a change to the algorithm that does not require B_+ to be biregular.

Appendix A


Chain Rule
$G \subseteq \mathfrak{S}(\Omega)$;
$\mathfrak{x}, \mathfrak{y} : \Omega \to \Sigma$.
Atkinson
Block-system $\mathcal{P}$ of G on Ω;
$H = \mathrm{im}(G \to \mathfrak{S}(\mathcal{P}))$
Is $|H|$ small?
yes
no. Babai, Luks, Seress
Set-wise stabiliser
$I \subseteq G$ of the $\mathcal{P}$.
Cameron scheme
$f : \mathcal{P} \leftrightarrow \binom{B}{t}^r$.
Weak Luks reduction
Going Down
Orbits $(\Delta_i)_i$ of G.
Giant representation
$\phi : G \to \mathfrak{S}(\Gamma)$;
Standard blocks Φ.
Is G primitive?
yes
no
$f_+ : \Omega \leftrightarrow \binom{B}{t}^r$
$|\Phi| \leq (1/2)|\Omega|$
TopAction
AggregateCertificates
Coloured α-partition or
k-ary configuration on Γ
Extended Design Lemma
Coloured α-partition or
embedded Johnson scheme on Γ
Going Up
Colouring $(\Delta_i)_i$ of Ω;
Updated G respecting the colouring;
Giant representation $\phi_+ : G_{\Delta_1} \to \mathfrak{S}(\Gamma_+)$

Appendix B

Several algorithms we considered in this thesis have been implemented in C++ by the author of this thesis. The source-code is hosted at the on-line git repository https://github.com/MadPidgeon/Babai-Graph-Isomorphism. The code compiles with GCC version 5.3.0. For compilation, simply call '`make all`'.

The folder `misc/` contains some functions with no particular significance in their ordering, and are characterised by having no theoretical value. The files there are mainly to get C++ to work.

The folder `examples/` contains examples to using the classes we define. For example, `examples/configurations.cc` checks whether a Chang graph truly induces a UPCC by applying Weisfeiler-Lehman refinement.

The files `permutation.cc`, `group.cc`, `coset.cc` and `action.cc` and their corresponding header files give the basic definitions of the group-theoretical structures that their names imply.

The file `unionfind.cc` is used by several parts of the algorithm to represent a partitioning of a set. Its main use is in `action.cc`, where it is used in the implementation of Theorem 2.18.

The file `fhl.cc` implements most of the algorithms from Section 2.3, most notably Theorem 2.11.

The file `luks.cc` implements Luks' part of Babai's string isomorphism algorithm, Algorithm 2.25.

The file `cameron.h` implements the recovery of the Cameron scheme, Algorithm 4.13. It depends on `multi.h` to set up multi-threading.

The file `datastructures.cc` defines objects the other categories we consider, like the coherent configuration and the bipartite graph. In this file, Proposition 6.9 is implemented.

The file `design_lemma.cc` implements the Design Lemma. Though this algorithm was not treated in this thesis, it is a good example of reductions at multiplicative cost (Section 3).

References

[1] László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015.

[2] Merrick Furst, John Hopcroft, and Eugene Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, SFCS '80, pages 36–41, Washington, DC, USA, 1980. IEEE Computer Society.

[3] M. D. Atkinson. An algorithm for finding the blocks of a permutation group. *Mathematics of Computation*, 29(131):911–913, 1975.

[4] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, April 1975.

[5] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42 – 65, 1982.

[6] Peter J. Cameron. Finite permutation groups and finite simple groups. *Bull. London Math. Soc*, 13:1–22, 1981.

[7] Attila Marti. On the orders of primitive groups. *Journal of Algebra*, 258(2):631 – 640, 2002.

[8] L. Babai, E. Luks, and A. Seress. Permutation groups in NC. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 409–420, New York, NY, USA, 1987. ACM.

[9] John D. Dixon and Brian Mortimer. *Permutation Groups*, chapter 5.2, pages 147–151. Springer New York, New York, NY, 1996.

[10] L. Babai, P. Erdős, and S. Selkow. Random graph isomorphism. *SIAM J. Comput.*, 9(3):628–635, 1980.

[11] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. Proc. 45th ACM STOC, pages 271–280, 2013.

[12] B.J. Weisfeiler and A.A. Leman. Reduction of a graph to a canonical form and an algebra arising during this reduction. *Naucho–Technicheskaya Informatsia*, 2(9):12–16, 1968.

[13] Neil Immerman and Eric Lander. *Describing Graphs: A First-Order Approach to Graph Canonization*, pages 59–81. Springer New York, New York, NY, 1990.

[14] L.C. Chang. Association schemes of partially balanced block designs with parameters $v = 28$, $n_1 = 12$, $n_2 = 15$ and $p_{11}^2 = 4$. *Sci. Record*, 4:12–18, 1960.

[15] Petteri Kaski and Patric Östergård. The steiner triple systems of order 19. *Mathematics of Computation*, 73(248):2075–2092, 2004.