Bounded Color Multiplicity Graph Isomorphism is in the #L Hierarchy*

V. Arvind, Piyush P Kurur, and T.C. Vijayaraghavan
Institute of Mathematical Sciences, C.I.T Campus
Chennai 600113, India
email: {arvind,ppk,tcvijay}@imsc.res.in

Abstract

In this paper we study the complexity of *Bounded Color Multiplicity Graph Isomorphism* BCGI_b: the input is a pair of vertex-colored graphs such that the number of vertices of a given color in an input graph is bounded by b. We show that BCGI_b is in the #L hierarchy (more precisely, the $\mathrm{Mod}_k\mathrm{L}$ hierarchy for some constant k depending on b). Combined with the fact that Bounded Color Multiplicity Graph Isomorphism is logspace many-one hard for every set in the $\mathrm{Mod}_k\mathrm{L}$ hierarchy for any constant k, we get a tight classification of the problem using logspace-bounded counting classes.

1 Introduction

In the last decade or so, logspace bounded classes, particularly logspace counting classes, have played an important role in classifying natural problems in NC^2. A well-known example is the problem of computing the determinant. We know from Toda and Vinay's work [Tod91, Vin91] that the complexity class GapL *exactly* captures the complexity of computing integer determinants. Likewise, computing determinants over finite fields of characteristic p is captured by the class $\mathrm{Mod}_p\mathrm{L}$ [BDHM92]. More recently, the results of [ABO99] classify important linear algebraic problems like finding the rank and checking feasibility of linear equations. Also, the complexity of perfect matching is now quite well characterized in [ARZ99] using logspace counting classes and the isolation lemma. For other results in this area, including connections to circuit complexity questions, we refer the reader to the recent survey article by Allender [Al04].

*Part of this work was supported by a DST-DAAD project supporting exchange visits.

1.1 Graph Isomorphism and Logspace Counting Classes

In this paper we study the following restricted version of Graph Isomorphism problem: The input is a pair of vertex-colored graphs (G_1, G_2) with the additional property that the number of vertices of a given color in the graphs is less than a prescribed constant bound b. And, the restricted Graph Isomorphism problem is to check if there is a color-preserving isomorphism between G_1 and G_2.

This problem is known as the *Bounded Color Multiplicity Graph Isomorphism* problem (which we denote by BCGI_b). It is a well-studied restriction of Graph Isomorphism: This was the first version of Graph Isomorphism studied using group-theoretic methods when Babai gave a randomized polynomial time algorithm for BCGI_b for each constant b [Bab79]. This result was improved to a deterministic polynomial time algorithm in [FHL80]. Subsequently, Luks in [Lu86] gave a remarkable NC algorithm for the BCGI_b problem. Building on earlier work by McKenzie and Cook [MC87]) and Luks and McKenzie [LM88], in [Lu86] Luks introduced group-theoretic techniques to handle the nonabelian parts of permutation groups in the class NC. The earlier papers [MC87, LM88] developed techniques to show that several permutation group problems for abelian and solvable groups are in the class NC. These methods are linear-algebraic and do not extend to general permutation groups.

Luks' NC algorithm in [Lu86] is actually for a more general problem (of which BCGI_b is a special case). In [Lu86] the actual parallel time taken by the algorithm is not specified. But we can notice that the NC algorithm runs in $O(\log n)$ stages and in each stage it uses Luby's NC^2 algorithm for the maximal independent set problem. Thus the running time of the parallel algorithm appears to be at least $\log^3 n$.

In a different line of research, Torán in [Tor04] developed a nice graph gadget that enables the simulation of addition modulo k using automorphisms of the graph gadget. As a consequence, in [Tor04] several hardness results for Graph Isomorphism are shown. In particular, it is shown that BCGI_b is AC^0-many one hard for the logspace counting class Mod_kL for each constant k. The construction in [Tor04] requires b to be k^2.

In a related paper [JKMT04], the BCGI problem is examined for color classes of sizes 2 and 3, and show that in these cases it is in SL (which puts it in L as $\text{SL} = \text{L}$ [Re04]).

1.2 The results of this paper

The question that naturally arises is whether the gap between the NC upper bound result of [Lu86] and the lower bound result of [Tor04] for BCGI_b can be closed. In this paper, we examine Luks' NC algorithm carefully using logspace complexity classes. As our main result we show that BCGI_b is in $\mathrm{Mod}_k\mathrm{L}$ hierarchy, where the constant k and the level of the hierarchy in which BCGI sits depends on b.

It turns out that the we can decompose the problem into a constant number of stages. Each stage corresponds to either handling an abelian simple group or a nonabelian simple group. We are able to show that the stages involving abelian groups can be solved in the complexity class $\mathrm{Mod}_k\mathrm{L}$ for a suitable constant k. Surprisingly for handling nonabelian groups, although the group theory is more involved, the actual computation can be done in deterministic logspace L. This is because in the nonabelian stages we can do away with finding maximal independent sets and show that the computations involved are actually reducible to *undirected graph connectivity* (which is recently shown to be in L [Re04]).

Since the $\mathrm{Mod}_k\mathrm{L}$ hierarchy is contained in the #L hierarchy which, in turn, is contained in NC^2 (even in TC^1) [ABO99], our result puts BCGI_b in NC^2 (indeed in TC^1).

Below we depict a chain known inclusions of the relevant complexity classes inside NC^2.

$$\mathrm{BCGI}_b \in \mathrm{Mod}_k\mathrm{L}^{\mathrm{Mod}_k\mathrm{L}^{\cdot^{\cdot}}} \subseteq \#\mathrm{L}^{\#\mathrm{L}^{\cdot^{\cdot}}} \subseteq \mathrm{TC}^1 \subseteq \mathrm{NC}^2.$$

While the ideas and the underlying group-theoretic machinery in our result is based on Luks [Lu86] and [LM88], to prove the $\mathrm{Mod}_k\mathrm{L}$ hierarchy upper bound we had to make several crucial changes in the NC algorithm of [Lu86].

Remark. *How well does our* $\mathrm{Mod}_k\mathrm{L}$ *hierarchy upper bound classify the problem* BCGI_b*?*

We can apply Torán's result [Tor04, Lemma 4.7] to show that for every k *if* A *is a set in the* j*th level of the* $\mathrm{Mod}_k\mathrm{L}$ *hierarchy, then* A *is logspace many-one reducible to* BCGI_b*, where* b *depends only on* k *and* j*. Thus,* BCGI_b *is intimately related to the* $\mathrm{Mod}_k\mathrm{L}$ *hierarchy. The question whether the* $\mathrm{Mod}_k\mathrm{L}$ *hierarchy collapses for composite* k *is open (for prime* k *it is known to collapse to* $\mathrm{Mod}_k\mathrm{L}$*).*

2 Preliminaries and Notation

We assume some familiarity with basic complexity theory and basic notions in group theory and linear algebra. For details on the complexity classes discussed here we refer the reader to [ABO99] and for group-theoretic definitions we refer the reader to [Ha, Wie64].

2.1 Complexity Classes

We write L and FL for the class of languages (respectively, functions) computable by deterministic logspace-bounded Turing machines. The class of languages accepted by nondeterministic logspace machines is denoted by NL, and the class of languages accepted by symmetric logspace machines is denoted by SL (equal to L in [Re04]). The class NL is known to be closed under complement.

We now recall the definitions for the complexity classes GapL and $\mathrm{Mod}_k\mathrm{L}$ with $k \geq 2$. For a nondeterministic Turing machine M we denote by $acc_M(x)$ and $rej_M(x)$ the number of accepting and rejecting computation paths of M on input x, respectively. We define $gap_M(x)$ as the difference $acc_M(x) - rej_M(x)$.

Definition 2.1. *The class* GapL *is defined as the class of functions* $f : \Sigma^* \longrightarrow \mathbb{Z}$ *such that for some nondeterministic logspace-bounded Turing machine* M, $f(x) = gap_M(x)$ *for every* $x \in \Sigma^*$.

GapL can also be defined equivalently as the closure of #L under subtraction, where #L denote the set of all functions $acc_M(x)$ for nondeterministic logspace-bounded Turing machines M.

Definition 2.2. *Let* $k \geq 2$ *be an integer.* $\mathrm{Mod}_k\mathrm{L}$ *is the class of sets* L *such that there is an* $f \in \#\mathrm{L}$ *with* $x \in L \Leftrightarrow f(x) \not\equiv 0 (mod\ k)$.

We now define the reductions that are discussed in this paper: for languages L and L', we say that L is *logspace many-one reducible* to L' if there is an $f \in \mathrm{FL}$ such that $x \in L$ if and only if $f(x) \in L'$. We say that L is *logspace Turing reducible* to L' if there is a deterministic logspace oracle Turing machine M such that M with oracle L' accepts L.

Nondeterministic logspace oracle Turing machines are defined using the "Ruzzo-Simon-Tompa"" oracle access [RST84] (also see [AO96, ABO99]) which restricts the machine to write the oracle queries deterministically. We define the relativized classes $\#\mathrm{L}^A$ and $\mathrm{Mod}_k\mathrm{L}^A$ in this model. We can now inductively define the $\mathrm{Mod}_k\mathrm{L}$ hierarchy: A language L is in the jth

level of the $\mathrm{Mod}_k\mathrm{L}$ hierarchy if there is an oracle set A in the $j-1$th level such that $L \in \mathrm{Mod}_k\mathrm{L}^A$.

For languages L and L', we say that L is NC^1-Turing reducible to L' if there is a logspace uniform circuit family of polynomial size and logarithmic depth with bounded fanin gates that computes L with oracle gates for L'. AC^0-Turing reductions are similarly defined using logspace uniform AC^0 circuit families. It is known that the #L hierarchy equals $\mathrm{AC}^0(\#\mathrm{L})$ [AO96].

2.2 Permutation Groups

We recall useful definitions and basic facts about permutation groups. For details we refer the reader to the book by Wielandt [Wie64].

A permutation group G is a subgroup of $\mathrm{Sym}\,(\Omega)$, where $\mathrm{Sym}\,(\Omega)$ is the group of all permutations on Ω, and Ω is an n-element set. We write $H \leq G$ when H is a subgroup of G. The subgroup H of G is *normal*, denoted $H \lhd G$, if $gHg^{-1} = H$ for all $g \in G$.

The image of a point $\alpha \in \Omega$ under $g \in G$ is denoted α^g, and we apply permutations from left to right so that $\alpha^{gh} = (\alpha^g)^h$. The set $\alpha^G = \{\alpha^g \mid g \in G\}$ is the *G-orbit* of α, and G is *transitive* on Ω if $\alpha^G = \Omega$. The *group generated* by a set of permutations $S \subseteq \mathrm{Sym}\,(\Omega)$ is the smallest subgroup of $\mathrm{Sym}\,(\Omega)$ containing S, denoted $\langle S \rangle$. For a group G and $S \subseteq G$, the *normal closure* of S in G is the smallest normal subgroup of G containing S and is denoted by $\mathrm{NCL}_G(S)$.

A group G is *cyclic* if $G = \langle g \rangle$ for some g, G is *abelian* if $gh = hg$ for all $g, h \in G$.

Let $G \leq \mathrm{Sym}\,(\Omega)$ be transitive on Ω. A subset $\Delta \subseteq \Omega$ is called a *G-block* if for every $g \in G$ either $\Delta^g = \Delta$ or $\Delta^g \cap \Delta = \emptyset$. For any transitive group G, the set Ω and the singleton sets $\{\omega\}$, $\omega \in \Omega$ are called *trivial* blocks. A transitive group G is called *primitive* if it does not have any nontrivial blocks otherwise it is called *imprimitive*.

Consider a permutation group G that is transitive on Ω. If Δ is a G-block, then for every $g \in G$ Δ^g is also a G-block. The collection of blocks $\{\Delta^g : g \in G\}$ forms a partition of Ω and is called the Δ *block system*. Notice that the permutation group G acts transitively on the Δ block system (since every element of G in its natural action maps blocks to blocks). We call $\Delta \subset \Omega$ a *maximal block* if there is no other block Σ such that $\Delta \subset \Sigma \subset \Omega$. In this case, we call $\{\Delta^g : g \in G\}$ a *maximal block system*.

If G's action is imprimitive on a maximal block system, then we can again find a maximal block system here and continue further, resulting in a hierarchy of block systems. We formalize this process below.

Let Δ and Σ be two G-blocks in Ω, such that $\Delta \subseteq \Sigma$. Notice that the collection of blocks $\{\Delta^g : g \in G \text{ and } \Delta^g \subseteq \Sigma\}$ forms a partition of Σ which we call the Δ *block system* of Σ.

We now define a structure tree of the transitive group G. Structure trees play an important role in algorithms for permutation groups [Luk93]. The *structure tree* of G is a rooted tree nodes of which are labeled by G-blocks satisfying the following conditions:

1. The root is labeled Ω.
2. The leaves are labeled with singleton sets $\{\alpha\}$, $\alpha \in \Omega$.
3. For each internal node labeled by Σ, the labels of its children constitute a Δ block system of Σ, where Δ is a maximal block properly contained in Σ.

Note that there is a natural action of G on nodes at each level: $g \in G$ maps node u to v iff there is a G-block Δ such that the labels of u and v are Δ and Δ^g respectively. Notice that G acts primitively on the children of the root.

Let G be a permutation group on Ω with orbits $\Omega_1, \ldots, \Omega_r$. The *structure forest* is a collection of structure trees $T_1, \ldots, T-r$, where T_i is the structure tree of the transitive action of G restricted to Ω_i.

3 Overview of the algorithm

Let $\Omega = \{1,\ 2, \ldots,\ n\}$ and $G \leq Sym(\Delta)$ be a group. The group $G_{\{\Delta\}}$ is the *pointwise stabilizer* of Δ, that is the subgroup of elements in G that fix each point in Δ.

We now define BCGI and a closely related problem POINTSET$_{\text{b}}$.

Definition 3.1.

- *The problem* BCGI$_b$ *has input instances as pairs of graphs* (G_1, G_2) *where the two graphs* G_1 *and* G_2 *are vertex colored such that each color class is of size at most* b*. The problem is to test if there is a color-preserving isomorphism between* G_1 *and* G_2*. We sometimes write BCGI to stand for the problem when the constant* b *is not specified.*

- *The problem* POINTSET$_{\text{b}}$ *has input instances consisting of a subset* $\Delta \subseteq \Omega$ *along with a permutation group* $G \leq Sym(\Omega)$ *given by its generator set, such that* G *has orbits of size bounded by a constant* b*. The problem is to find the pointwise stabilizer subgroup* $G_{\{\Delta\}}$ *of* G*.*

It is well known that for each constant b there is a constant c such that BCGI_b reducible to POINTSET_c [LM88, Lu86]. If we examine the reduction it is easy to see that it is indeed a logspace many-one reduction. Thus we have the following proposition.

Proposition 3.2. BCGI_b *is logspace many-one reducible to* POINTSET_c*, where the constant c depends on the size of the color classes b.*

Thus, in order to prove the upper bound result for BCGI, it suffice to show that the problem POINTSET_b is in the Mod_kL hierarchy for some constant k. The rest the paper is devoted to this. We first give an overview of the algorithm, breaking it up into different tasks. In the subsequent sections we describe algorithms for these tasks in detail.

Let $\Omega = \cup_{i=1}^{m}\Omega_i$ be the partition of Ω into G-orbits, where $|\Omega_i| \leq b$ for each i. Let G_i be the permutation group obtained by projecting G onto Ω_i for each i. Since $|G_i| \leq b!$ for each i, the following proposition is immediate.

Proposition 3.3. *Given an instance* $G \leq Sym(\Omega)$ *of* POINTSET_b*, the orbits* Ω_i *and the groups* G_i *obtained by projecting* G *on* Ω_i *can be computed in deterministic logspace.*

Furthermore, for each G_i(acting transitively on Ω_i) we can compute a structure tree of G_i in deterministic logspace: we can simply search for a minimal block B_1 and find the corresponding block system by letting G_i act on it. Thus we obtain $\Omega_i = B_1 \cup \cdots \cup B_l$, where $\{B_i\}$ is the block system. Either G_i acts primitively on $\{B_1, \ldots, B_l\}$ or we can again find a minimal block and a corresponding block system. Continuing thus, we finally get a list of blocks on which G_i acts primitively. We get a rooted tree corresponding to the above algorithm: the leaves are elements of Ω_i, the next level has l nodes corresponding to each of $B_1, \ldots, B_l$ and, the children of B_j are the elements of Ω_i it contains and so on. At the k^{th} level of the tree are the nodes corresponding to a minimal block system for G_i's action on the block system at level $k-1$.

Proposition 3.4. *For each orbit of* G*, an instance of* POINTSET_b *a structure tree can be computed in deterministic logspace.*

Since the orbit sizes are constant, the above proposition is immediate.

Let $T_1, \ldots, T_m$ be the computed structure trees for the respective orbits $\Omega_1, \ldots, \Omega_m$ of G. Every T_i has constant height (bounded by $\log b$). Notice that $G_{\{\Delta\}}$, the subgroup we need to compute, fixes every ancestor of points in Δ of a structure tree T_i, $(1 \leq i \leq m)$. This property will allow us to reduce the problem further as follows:

Consider G's action on the set $\Omega^{'} = \cup_{i=1}^{m} \Omega_i^{'}$, where $\Omega_i^{'}$ is the set of children of the root of T_i, $(1 \leq i \leq m)$. By definition G acts primitively on each $\Omega_i^{'}$.

Let $\Delta^{'} \subseteq \Omega^{'}$ be the ancestors of Δ in all the structure trees T_i. Then $(G,\ \Delta^{'},\ \Omega^{'})$ is an instance of POINTSET$_\text{b}$ with the further restriction that G is primitive on its orbits. We refer to this restricted problem as primitive-POINTSET$_b$. Suppose we have an algorithm for primitive-POINTSET$_b$ in the Mod$_k$L hierarchy for some constant k. This algorithm will compute $G_{\{\Delta'\}} = H$. We will assume that the algorithm will actually compute a generating set $\{h_1, \ldots,\ h_s\}$ for H, where it keeps track of h_i's action on the entire set Ω.

Notice that the group $G_{\{\Delta\}}$ is a subgroup of H. Now consider the structure forest of H. For $\delta \in \Delta$ let T_δ be the tree with δ as a leaf in the structure forest for H. It is easy to see that the height of T_δ is 1 less than the height of the corresponding tree in the structure forest for G.

Thus, in computing H we have made progress. We can replace G by H, and recompute the structure forest for H and repeat the above. If we invoke the Mod$_k$L hierarchy upper bound for primitive-POINTSET$_b$ a constant number of nested oracle levels, where the constant is bounded by $\log b$, then we would have computed $G_{\{\Delta\}}$.

Putting it together, we have shown the following reduction.

Theorem 3.5. *If primitive-POINTSET$_b$ is in the* Mod$_k$L *hierarchy then* POINTSET$_\text{b}$ *is also in the* Mod$_k$L *hierarchy.*

Remark. *Both the level of the hierarchy and k are constants that depend on the orbit bound b. We do not make it explicit here.*

Thus the problem that we need to upper bound is primitive-POINTSET$_b$. In the remaining sections we will show that this problem is in the Mod$_k$L hierarchy for a constant k.

4 Basic group theoretic lemmas

For proving our upper bound on primitive-POINTSET$_b$ we need to recall some group-theoretic machinery from [Lu86]. We shall also need to assemble specific properties of permutation groups with bounded size orbits.

Let $G \leq \text{Sym}\,(\Omega)$ and let Ω_i, $1 \leq i \leq m$ be its orbits such that $|\Omega_i| \leq b$ for all i. Let G_i be the projection of G onto Ω_i. Clearly, $G \leq G_1 \times \ldots \times G_m$. For any $h \in G_1 \times \ldots \times G_m$ let $pr_i\,(h)$ mean the action of h restricted to Ω_i.

For a subgroup H of $G_1 \times \ldots \times G_m$, we will use $pr_i(H)$ to denote projection of the subgroup H on Ω_i.

We have $G_i \leq \mathrm{Sym}(\Omega_i)$ and $|\Omega_i| \leq b$ for each i, the set of possible simple subgroups that can occur as quotients of consecutive groups in any composition series of $G_1 \times \ldots \times G_m$ is a constant sized collection $\mathcal{T} = \{T_1, \ldots, T_c\}$ where $c \leq 2^{b!}$. A deterministic logspace machine can explicitly compute and keep $\mathcal{T}$ as a table.

Definition 4.1 (T-semisimple groups)**.** *A finite group G is* T-semisimple*, for a simple group T, if $G = G_1 \times \ldots \times G_r$ where each G_i is isomorphic to T.*

Lemma 4.2. [Lu86] *For any simple group T and any finite group G, there is a unique minimal normal subgroup N such that G/N is T-semisimple.*

We call N as the T-residue of G and denote it by $\mathrm{Res}_T(G)$. By Lemma 4.2, notice that $\mathrm{Res}_T(G)$ is invariant under any automorphism of G.

Definition 4.3. *A subgroup H of a finite group G is a* characteristic subgroup *if H is invariant under all automorphisms of G.*

We recall two useful properties.

Proposition 4.4. *1. If H is a characteristic subgroup of G then H is a normal subgroup of G.*

2. If K is a characteristic subgroup of H and H is a characteristic subgroup of G then K is a characteristic subgroup of G.

For any simple group T, notice that $\mathrm{Res}_T(G)$ is a characteristic subgroup of G.

Definition 4.5. *A* residual tower *of G is a normal series $G = R_0 \rhd R_1 \ldots \rhd R_l = 1$ where for all $1 \leq i \leq l$, $R_i = \mathrm{Res}_{T_i}(R_{i-1})$ for some simple group T.*

Using Proposition 4.4 each R_i is a characteristic subgroup of G and hence is normal.

Definition 4.6. *The* socle $\mathrm{Soc}(G)$ *of a finite group G is the subgroup generated by the minimal normal subgroups of G.*

Proposition 4.7. *The socle* $\mathrm{Soc}(G)$ *is a characteristic subgroup of G.*

The following lemma is a crucial property of residual towers for primitive groups.

Lemma 4.8. [Lu86] *If H is a primitive permutation group acting on Ω then the smallest nontrivial group in any residual tower is always* $\mathrm{Soc}\,(H)$.

Now we consider an instance (G, Δ, Ω) of primitive-POINTSET$_b$. As the orbits Ω_i are of constant size, we can in deterministic logspace compute a residual tower $\{R_{i,j}\}_{j=1}^{l}$ for each G_i. We assume, without loss of generality, that the length l of residue sequences of G_i's are all same. Notice that l is a constant bounded by $\log b$.

We describe a series of groups for G obtained from these residual towers.[1]

First consider the groups $S_j = R_{1,j} \times \ldots \times R_{m,j}$. Notice that $S_0 = G_1 \times \ldots \times G_m$. Furthermore, since $R_{i,j} \lhd G_i$ for $1 \leq i \leq m$ and $1 \leq j \leq l$, it follows easily that we have a normal series

$$1 = S_l \lhd S_{l-1} \lhd \ldots \lhd S_1 \lhd S_0 = G_1 \times \ldots \times G_m.$$

We refine this normal series further as follows. Consider the list of simple groups $\{T_1, \ldots, T_c\}$ as an ordered list, we insert a series of length at most c between S_j and S_{j+1} for each j.

$$S_j = Q_0 \rhd Q_1 \rhd Q_2 \ldots Q_c = S_{j+1}.$$

where, for $1 \leq k \leq c$, the $Q_k = R_{1,j}^{(k)} \times \ldots \times R_{m,j}^{(k)}$ is given by

$$R_{i,j}^{(k)} = \begin{cases} R_{i,j+1} & \text{if } R_{i,j}/R_{i,j+1} \cong T_k \\ R_{i,j}^{(k-1)} & \text{otherwise} \end{cases}$$

Notice that each Q_k is a normal subgroup of S_0. Furthermore Q_k/Q_{k+1} is either trivial or is T_k-semisimple. As a result we have a normal series of length at most cl (a constant) of the following form

$$G_1 \times \ldots \times G_m = R_0(G) \rhd R_1(G) \rhd \ldots \rhd R_k(G) = 1, \tag{1}$$

where $R_i(G)/R_{i+1}(G)$ is T-semisimple for some $T \in \{T_1, \ldots T_c\}$.

Taking $N_i = G \cap R_i(G)$ we have the following *residue series*

$$G = N_0 \rhd \ldots \rhd N_k = 1, \tag{2}$$

where N_i/N_{i+1} can be embedded $R_i(G)/R_{i+1}(G)$. It follows that N_i/N_{i+1} is also T-semisimple for some $T \in \mathcal{T}$. Furthermore, $N_i \lhd G$ for all i.

We need the following properties of this series (as in [Lu86]).

[1] This is different from the series used in [Lu86], and is more convenient for proving the results of this paper.

Lemma 4.9. [Lu86] *Let N be a group in the normal series of Equation 2. Then for any subgroup $H \leq G$ the projection $pr_i(H) = G_i$ if and only if $pr_i(H \cap N) = pr_i(N)$.*

The following characterization of the socles of primitive permutation groups is also needed.

Theorem 4.10 (O'Nan-Scott theorem)**.** *Let $G \leq \mathrm{Sym}(\Omega)$ be a primitive permutation group with socle $\mathrm{Soc}(G) = K$. Let $\omega \in \Omega$. Then K is T-semisimple for some simple group G and one of the following holds:*

(i) K is abelian in which case K is elementary abelian, regular on Ω, and is the unique minimal normal subgroup of G. Furthermore $K_{\{\omega\}} = 1$.

(ii) K is nonabelian, transitive on Ω, and is the unique minimal normal subgroup of G.

(iii) K is nonabelian and is a product $K = K_1 \times K_2$, where K_1 and K_2 are isomorphic. K_1 and K_2 are the only minimal normal subgroup of G, and each K_i acts regularly on Ω. For each $\omega \in \Omega$, $K_{\{\omega\}} = \mathrm{Diag}(K_1 \times K_2)$.

Another crucial result is Scott's lemma as stated in [Lu86]. We state a version useful for us. Let G and H be finite isomorphic groups. Then for an isomorphism φ from G to H, the diagonal subgroup $\mathrm{Diag}(G \times H)$ of $G \times H$ defined as follows

$$\mathrm{Diag}(G \times H) = \{(x, \varphi(x)) \mid x \in G\}.$$

Clearly, $\mathrm{Diag}(G \times H)$ is isomorphic to each of G and H. If $\{H_i \mid 1 \leq i \leq r\}$ is a set of mutually isomorphic finite groups, we can similarly define $\mathrm{Diag}(\times_{i=1}^{r} H_i)$ as a subgroup of $\prod_{i=1}^{r} H_i$, given isomorphisms from H_1 to each of the H_i, $i \neq 1$.

Lemma 4.11 (Scott's Lemma)**.** *Let T be a finite simple group and let G be any subgroup of $\prod_{i=1}^{r} T_i$ such that each T_i is isomorphic to T. Then G is a direct product of diagonal subgroups. More precisely, there is a partition $\cup_{j=1}^{s} I_s$ of $\{1, \ldots, r\}$ such that*

$$G = \prod_{j=1}^{s} \mathrm{Diag}\left(\times_{i \in I_j} T_i\right).$$

We now give a brief overview of the algorithm for primitive-POINTSET$_b$. Let (G, Δ) be an instance of primitive-POINTSET$_b$ and let $G = N_0 \rhd \ldots \rhd N_l$ be the residue series for G as defined in Equation 2. The algorithm claimed in the following theorem is the key subroutine for solving primitive-POINTSET$_b$.

Theorem 4.12. *Given a instance (G, Δ) of primitive-POINTSET$_b$ and a residue series $G = N_0 \rhd \ldots \rhd N_l = \{1\}$ for the group G, there is a sequence of subgroups $G = H_0, H_1, \ldots, H_l$ of G such that each H_s in the sequence contains $G_{\{\Delta\}}$, and $|pr_i(H_s)| \leq |G_i|/2$ for all indices i such that $pr_i(N_s) = \{1\}$ and Ω_i contains a point of Δ. Furthermore, for each $s \geq 0$, given the subgroup H_s of this sequence by a generating set the subgroup H_{s+1} can be computed by an $\mathrm{L}^{\mathrm{Mod}_t\mathrm{L}}$ algorithm.*

For an instance (G, Δ) of primitive-POINTSET$_b$, call an orbit Ω_i a *target orbit* if Ω_i contains a point of Δ.

We compute the sequence of subgroups $H_0, \ldots, H_l$ as follows: To begin with $H_0 = G$. Then, H_{i+1} is computed from H_i by applying the $\mathrm{L}^{\mathrm{Mod}_t\mathrm{L}}$ algorithm of Theorem 4.12.

As a consequence of this theorem, the subgroup H_l is such that for all target orbits Ω_i, $|pr_i(H_l)| \leq |G_i|/2$. Thus, if we consider the group $H = H_l$, then notice that $G_{\{\Delta\}} \leq H$, and H has been cut down on all target orbits. We now recompute the residue series with G replaced by $H = H_l$ and repeat the above process. Since in every such iteration the group is cut down in size on each target orbit, in a *constant* number of rounds the algorithm will have converged to $G_{\{\Delta\}}$. This requires only a constant number of iterations as each G_i is a constant-sized group. We will prove Theorem 4.12 in Section 6.

For putting primitive-POINTSET$_b$ in the Mod$_k$L hierarchy we first need to compute the residue series of G for an instance (G, Δ) of primitive-POINTSET$_b$. In the next section we explain how to compute it in the Mod$_k$L hierarchy using a general notion of strong generating sets as in [LM88]. Combined with the algorithm sketched above this will prove the upper bound for primitive-POINTSET$_b$.

5 The Strong Generating Set problem

Let $G \leq \mathrm{Sym}(\Omega)$, where G has orbits $\Omega_i, 1 \leq i \leq m$ of size bounded by a constant b. Consider the *residue series* for G defined by Equation 2. In this section we show that computing a generating set for each subgroup in the residue series is in the Mod$_k$L hierarchy. The method we use is via strong

generating sets and sifts, as invented by Luks and McKenzie [LM88, Lu86]. We begin with a general definition.

Definition 5.1 (Strong generator set)**.** *Let G be a permutation group and let H be a subgroup of G. For a tower of subgroup $G = G_0 \geq \ldots \geq G_r = H$ let B_i, $0 \leq i < r$, be the set of all left coset representatives of G_{i+1} in G_i. Every element $g \in G$ can be uniquely expressed as a product $g = g_0 g_2 \ldots g_{r-1} h$ where $g_i \in B_i$ and $h \in H$. The collection of sets B_i, $0 \leq i < r$ is called a* strong generating set *of G relative to H (hereafter abbreviated as SGS for G rel H). If $H = 1$ then $\cup B_i$ is a generating set for G which is called a strong generating set for G.*

Given an element $g \in G$ and an SGS (say, $\cup B_i$) of G rel H, there is a unique $h \in H$ such that g can be written as a product $g = g_0 g_1 \ldots g_{r-1} h$ where $g_i \in B_i$. By the *sift* of g, denoted by $\mathrm{Sift}(g)$, we mean this element h. Note that the sift of an element depended on the choice of coset representatives B_i in the SGS of G.

We are interested in the special case when $H \lhd G$. For an SGS of G rel H arising from a *normal series* between G and H, we have the following lemma.

Lemma 5.2. *Let G be a permutation group and H be any normal subgroup of G. Let $G = G_0 \rhd G_1 \rhd \ldots \rhd G_k = H$ be a normal series between G and H and let B_i's be the left (right) coset representatives of G_{i+1} in G_i. Let $A = \{g_1, g_2, \ldots, g_l\}$ be a set of generators for G Let S be the set consisting of the following elements*

1. $\mathrm{Sift}(g)$ *for all $g \in A$.*
2. $\mathrm{Sift}(x^{-1}yx)$ *for all $x \in B_i$ and $y \in B_j$, $i \leq j$.*
3. $\mathrm{Sift}(xy)$ *for all $x, y \in B_i$ for all i.*

Then $\mathrm{NCL}_G(S) = H$.

Proof. Since H is a normal subgroup of G and since all elements of S are contained in H it is clear that $\mathrm{NCL}_G(S) \leq H$. To prove the converse consider any element $h \in H$. There exists elements $y_1, y_2, \ldots, y_m$ in A such that $h = y_1 y_2 \ldots y_l$. Now since S contains the sifts of all y's we have

$$h = \prod_{i=1}^{m} x_{i,1} x_{i,2} \ldots x_{i,r} s_i.$$

where $x_{i,j} \in B_j$ and s_i in $\mathrm{NCL}_G(S)$. Using the fact that $\mathrm{NCL}_G(S)$ is a normal subgroup and using properties 2 and 3 repeatedly we can rewrite the above product as $h = x_1 x_2 \dots x_r s$ where $x_i \in B_i$ and $s \in \mathrm{NCL}_G(S)$. Now since $h \in H$ we have $x_1 x_2 \dots x_r = 1$ and hence $h = s$. ■

5.1 Computing the Residue series

Let $G \leq \mathrm{Sym}\,(\Omega)$ with bounded orbits, given by a set of generators A. Let $\Omega_1, \dots, \Omega_m$ be its orbits of size at most b. Consider the residue series

$$G = N_0 \rhd N_1 \rhd \cdots \rhd N_k = \{1\}.$$

Recall that k is a constant depending on b, and each quotient N_i/N_{i+1} is T-semisimple for some simple group $T \in \{T_1, \dots, T_c\}$, where c is another constant depending on b.

The algorithm proceeds in stages: at the ith stage, we assume that an SGS for G rel N_i is already computed, and the task at this stage is to extend it to an SGS for G rel N_{i+1}. To do this, following the approach of [LM88], we will compute an SGS for N_i rel N_{i+1} and append it to the SGS for G rel N_i. If we can show that this step can be carried out in $\mathrm{L}^{\mathrm{Mod}_t\mathrm{L}}$ for a suitable constant t, the claimed $\mathrm{Mod}_k\mathrm{L}$ hierarchy upper bound will follow (where the level of the hierarchy is the number of stages).

In order to compute an SGS for N_i rel N_{i+1}, notice that it suffices to solve the problem defined in the theorem below (we can choose, in the theorem statement below, $L = R_i(G)$ and $M = R_{i+1}(G)$ as defined in Equation 1).

Theorem 5.3. *Let $G \leq \mathrm{Sym}(\Omega)$ with orbits Ω_i, $1 \leq i \leq m$ given by generator set A. Let $L = L_1 \times \ldots \times L_r$ and $M = M_1 \times \ldots \times M_r$ be such that $\mathrm{Sym}\,(\Omega_i) \geq L_i \geq M_i$ and G normalizes both L and M and L_i/M_i is T-semisimple for some simple group T. Given a set S of elements of G such that $\mathrm{NCL}_G(S) = G \cap L$ there is an $\mathrm{L}^{\mathrm{Mod}_t\mathrm{L}}$ algorithm to compute an SGS for $G \cap L$ rel $G \cap M$, for some constant t.*

Furthermore, for any $x \in G \cap L$ there is an $\mathrm{L}^{\mathrm{Mod}_t\mathrm{L}}$ algorithm to compute the $\mathrm{Sift}(x)$ w.r.t. the SGS for $G \cap L$ rel $G \cap M$ computed above.

Proof. The group L/M is T-semisimple as each L_i/M_i is T-semisimple. Consequently, L/M is of the form $T_1 \times T_2 \times \ldots T_s$ where $T_i \cong T$ for all $1 \leq i \leq s$, for some $s = O(m)$, where the constant factor depends on b. Moreover, $G \cap L/G \cap M$ can be faithfully embedded into $\prod_{i=1}^{m} L_i/M_i$.

We handle the proof of the theorem in two cases:

T is nonabelian.

By Scott's Lemma 4.11 we know that $G \cap L/G \cap M$ is a product of diagonal groups, each isomorphic to T. Let $\phi_i : L \mapsto T_i$ be the map obtained by composing the quotient map from L to L/M and the projection map to T_i. We say that T_i and T_j are *linked* if $G \cap L/G \cap M$ restricted to $T_i \times T_j$ is the diagonal group. Consider an undirected graph $\mathcal{G}$ with vertex set $V = \{T_i : 1 \leq i \leq t\}$ and edge set $\{(T_i, T_j) : T_i \text{ and } T_j \text{ are linked}\}$. The groups T_i and T_j are of bounded size hence we can check whether T_i and T_j are linked in deterministic logspace as follows:

Find $G \cap L/G \cap M$ restricted to $T_i \times T_j$. Since this is a constant size group one can easily determine whether the group is $T_i \times T_j$ or $\mathrm{diag}(T_i \times T_j)$ (by checking the order for example). To find $G \cap L/G \cap M$ we first compute the set $\phi_{ij}(S) = \{\langle \phi_i(s), \phi_j(s) \rangle : s \in S\}$. We keep on adding new elements by taking the G conjugates of elements of S and restricting it to the T_i and T_j until no more elements can be added. This will require only a constant number of such additions, and hence can be done in logspace.

Now in the graph $\mathcal{G}$ we find the connected components (in $\mathrm{L}^{\mathrm{SL}} = \mathrm{L}$). Let the connected components be C_k, $1 \leq k \leq l$. For each connected component C_k of $\mathcal{G}$ we pick a vertex, say T_{i_k}.

The main algorithmic step is the computation of elements $g_k \in G \cap L$, $1 \leq k \leq l$ such that $\phi_{i_j}(g_k) \in T_{i_j}$ is identity for all j except $j = k$. Assuming that we have these elements, consider the normal subgroup $\mathrm{NCL}_G(g_k)$ of $G \cap L$. Since ϕ_{i_k} is an onto homomorphism and T_{i_k} is simple, it follows that $\phi_{i_k}(\mathrm{NCL}_G(g_k))$, being a nontrivial normal subgroup of T_{i_k}, must in fact be T_{i_k}. We compute a set B_k of distinct inverse images of $\phi_{i_k}(\mathrm{NCL}_G(g_k))$, $1 \leq k \leq l$. It is easy to see that $\cup_{k=1}^{l} B_k$ is an SGS of $G \cap L$ rel $G \cap M$.

In the sequel, we explain how to compute the g_k's. It suffices to explain how to compute g_1.

To begin with we define *iterated commutators*. Let $[h_1, \ldots, h_k]$ be defined as

$$\begin{aligned} [h_1, h_2] &= h_1^{-1} h_2^{-1} h_1 h_2, \\ [h_1, \ldots, h_i, h_{i+1}] &= [[h_1, \ldots, h_i], h_{i+1}]. \end{aligned}$$

We will compute a sequence of elements $h_1, h_2, \ldots, h_l$ satisfying the following property.

1. $\phi_{i_1}(h_1) \neq 1$,
2. $\phi_{i_1}([h_1, h_2, \ldots, h_k]) \neq 1$ for all k and

3. $\phi_{i_k}(h_k) = 1$.

It is easy to see that given $h_1, h_2, \ldots, h_l$ as above, $g_1 = [h_1, h_2, \ldots, h_l]$ has the required properties: $\phi_{i_1}(g_1) \neq 1$, and $\phi_{i_k}(g_1) = 1$ for $2 \leq k \leq l$.

We now give a logspace algorithm to actually find such a sequence $h_1, h_2, \ldots, h_l$.

$x := h_1$ such that $\phi_{i_1}(h_1) \neq 1$;
$\alpha := \phi_{i_1}(h_1)$;
for $k = 2$ **to** l **do**
  compute h_i such that $\phi_{i_1}(h_k)$ does not commute with α and
   $\phi_{i_k}(h_k) = 1$;
  $\beta := \phi_{i_1}(h_1)$;
  $\alpha := [\alpha, \beta]$;
  output h_k;
end

Having got the sequence $h_1, \ldots, h_l$ we now need to compute the iterated commutator $[h_1, \ldots, h_l]$ in logspace. Then, combining the two logspace computations we get a logspace algorithm for computing g_1.

Claim. *Let G be a permutation group with bounded-size orbits. Given $h_1, \ldots, h_n \in G$, the iterated commutator $[h_1, \ldots, h_n]$ can be computed in deterministic logspace.*

It suffices to compute $[h_1, \ldots, h_n]$ restricted to each G-orbit Ω separately. Since G has bounded-size orbits, and since the iterated commutator $[h_1, \ldots, h_n]$ is a formula over the h_i's, we can evaluate $[h_1, \ldots, h_n]$ restricted to each orbit Ω_i in NC^1 (and hence deterministic logspace). The claim follows.

Finally, we claim that B_k can be computed in deterministic logspace from g_k. This is possible because it suffices to carry out the normal closure restricted to one of orbits on which g_k is nontrivial. However, notice that the computation will obtain the elements of B_k as permutations on the whole set Ω. There will be only a constant number of elements to be included in each B_k. Thus, we have computed $\cup_{k=1}^{l} B_k$ which is an SGS of $G \cap L$ rel $G \cap M$.

We now explain how to compute Sift(x) for any $x \in G \cap L$ w.r.t. this SGS:

Let $x \in G \cap L$ to be sifted. We apply $\phi_{i_k}(x)$ which is in T_{i_k} for each k, corresponding to the vertices T_{i_k} picked from each connected component C_k

of $\mathcal{G}$, $1 \leq k \leq l$. Since $\phi_{i_k}(B_k) = T_{i_k}$, for each k, in logspace we can find an x_k in the SGS such that $\phi_{i_k}(x_k) = (\phi_{i_k}(x))^{-1}$. It is now easy to see that $\text{Sift}(x) = x \prod_{k=1}^{l} x_k$.

This completes the proof for the nonabelian T.

T is abelian.

As T is abelian and simple, for some prime $p \leq b$, T is isomorphic to the additive group $\mathbb{Z}_p$. Since L/M to be T-semisimple, we have $L/M = T_1 \times T_2 \times \ldots T_s$ where $T_i \cong \mathbb{Z}_p$ for each i, and $s = O(m)$. Thus, L/M is essentially $\prod_{i=1}^{m} \mathbb{Z}_p$. But $\prod_{i=1}^{m} \mathbb{Z}_p$ has more structure because it is an m-dimensional vector space over the finite field $\mathbb{F}_p$. Furthermore, since $G \cap L/G \cap M$ can be faithfully embedded in $\prod_{i=1}^{m} L_i/M_i$, we know that $G \cap L/G \cap M$ is essentially a subspace of $\prod_{i=1}^{m} \mathbb{Z}_p$.

Notice that the vector space structure of $L/M = \prod_{i=1}^{m} \mathbb{Z}_p$ is effectively obtained: we have the generating set of $L/M = \prod L_i/M_i$, say $\{s_1 M, \ldots, s_r M\}$, in terms of the generating sets for L_i/M_i, where we know the permutations s_i on the entire set Ω. Thus, in deterministic logspace we can convert an element sM of L/M into the corresponding vector $\mathbf{v}_s \in \prod_{i=1}^{m} \mathbb{Z}_p$. Although the computation of SGS for $G \cap L$ rel $G \cap M$ and the Sift operation will be solved as a linear algebra problem modulo p, alongside we need to compute the SGS for $G \cap L$ rel $G \cap M$ in as elements of $\text{Sym}\,(\Omega)$. Similarly, we need to do the Sift operation on elements of $\text{Sym}\,(\Omega)$. Thus, it is important for the algorithm to keep track of the permutation π corresponding to each vector $\mathbf{v}_\pi$ that is computed. However, notice that this is easy to do because for $\mathbf{v}_\pi$ expressed as an $\mathbb{F}_p$ linear combination of the $\mathbf{v}_{s_i}$'s. More precisely, we know that if s and s' correspond to $\mathbf{v}_s$ and $\mathbf{v}_{s'}$, then ss' corresponds to $\mathbf{v}_s + \mathbf{v}_{s'}$ and s^k corresponds to $k\mathbf{v}_s$ for $k \in \mathbb{Z}_p$.

Now, given $S \subset G$ such that $\text{NCL}_G(S) = G \cap L$, we will show how to compute an SGS for $G \cap L$ rel $G \cap M$ in $\text{FL}^{\text{Mod}_p\text{L}}$. We need the following observation from [LM88] about the normal closure that will convert the problem to a simple linear algebra problem over $\mathbb{F}_p$ that can be solved in $\text{FL}^{\text{Mod}_p\text{L}}$ [ABO99, BDHM92].

Define $\tau_g : L/M \to L/M$ as $\tau_g(h) = g^{-1}hg$ where g is a generator of G and $h \in L/M$. The mapping τ_g is clearly an automorphism of the group L/M. Since L/M is the vector space $\prod_{i=1}^{m} \mathbb{Z}_p$, τ_g is an invertible linear transformation for each $g \in G$. Furthermore, given $g \in G$ in deterministic logspace we can write down the $m \times m$ matrix t_g corresponding to τ_g. It is easy to see that the matrix t_g will actually be a block diagonal matrix, with one block for each L_i/M_i, $1 \leq i \leq s$. The logspace algorithm just needs

to figure out the block corresponding to L_i/M_i for each i. Since L_i/M_i are constant sized, this can be easily done by exhaustive search.

Thus, if $G = \langle g_1, \ldots, g_q \rangle$, then in logspace we can compute the list of matrices $t_{g_1}, \ldots, t_{g_q}$. Each matrix t_{g_i} here is block diagonal, with each block being a square matrix of size at most c (where c is a constant depending on b). The following claim is from [LM88].[2]

Claim. $\mathrm{NCL}_G(S)/G \cap M$ *is the smallest subspace of* L/M *that contains the set of vectors* S *and is closed under the linear transformations* $\tau_{g_1}, \ldots, \tau_{g_q}$.

Thus, the problem of computing the SGS for $G \cap L$ rel $G \cap M$ boils down to finding the smallest subspace of $\prod_{i=1}^{m} \mathbb{Z}_p$ containing the vectors $\{\mathbf{v}_s \mid s \in S\}$ and closed under the linear maps defined by $t_{g_1}, \ldots, t_{g_q}$. Since each matrix t_{g_i} is block diagonal with each block of size bounded by the constant c, the matrix algebra $\mathcal{A}$ that is generated by $t_{g_1}, \ldots, t_{g_q}$ can be characterized by the following claim.

Claim. *Let* $\mathcal{M}$ *consist of the set of products of all choices of* j *matrices from* $\{t_{g_1}, \ldots, t_{g_q}\}$, *for each* j *such that* $1 \leq j \leq c^2$. *Then the matrix algebra* $\mathcal{A}$ *is the* $\mathbb{F}_p$ *linear span of the set* $\mathcal{M}$.

The above claim follows easily from the fact that the subalgebra generated by each block can have dimension at most c^2. Thus, it suffices to consider at most c^2 many matrix products for all the blocks to obtain an $\mathbb{F}_p$ linear basis for $\mathcal{A}$.

With the above claim the $\mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$ algorithm is now easy to describe. First, a deterministic logspace algorithm can output the set $\mathcal{M}$, since multiplying a constant number of matrices can be done in logspace. For each matrix $t \in \mathcal{M}$, notice that the logspace algorithm can also keep track of the corresponding element $g \in G$, by carrying out the corresponding multiplication.

Next, a deterministic logspace algorithm can list out the vectors $t(\mathbf{v}_s) \mid s \in S, t \in \mathcal{M}\}$. The corresponding elements as permutations of the form $g^{-1}sg$ can be kept track of again. To obtain the SGS from this set of vectors, we need to pick a maximal linearly independent subset. Since testing feasibility of linear equations over $\mathbb{F}_p$ can be done in $\mathrm{Mod}_p\mathrm{L}$ [ABO99], it follows that a basis (and hence an SGS) can be computed in $\mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$. The

[2] In [LM88] this is used for showing that the membership testing problem over permutation groups is in NC. We have adapted this to the BCGI setting where we crucially use the fact that the matrices are block diagonal with constant-size blocks to derive our upper bound.

corresponding permutations are also available. Thus, we have the SGS both as permutations and in vector form. Let the SGS as permutations be denoted by the list $\{\pi_1, \ldots, \pi_t\}$.

Finally, we need to describe $\text{Sift}(g)$ for any $g \in G$ using this SGS. We first write g as a vector $\mathbf{v}_g$ in L/M in logspace (by examining the action of g on each orbit). We write down a system of linear equations $AX = \mathbf{v}_g$, where the ith column A_i of A is $\mathbf{v}_{\pi_i}$ for $1 \leq i \leq t$, and π_i are the elements of the SGS. We compute the unique solution $x_1, \ldots, x_r$ to this in $\text{FL}^{\text{Mod}_p\text{L}}$. Notice that $\text{Sift}(g) = g \prod_{i=1}^{r} g_i^{-x_i}$. ■

Now we are ready to give the overall algorithm for computing the SGS of a permutation group with bounded orbits via its residue series.

Theorem 5.4. *Let $G \leq \text{Sym}(\Omega)$ be a permutation group with orbits Ω_i of size bounded by a constant such that $pr_i(G)$ is primitive on Ω_i for each i. Then an SGS (strong generating set) for G w.r.t. its residue series can be computed in the* Mod_kL *hierarchy.*

Proof.

Let G_i be the projection $pr_i(G)$ of G onto the ith orbit Ω_i. Recall from Equation 2 that the residue series of G is

$$G = N_0 \rhd \ldots \rhd N_k = 1,$$

which the groups N_i are defined w.r.t. the other series given by Equation 1:

$$G_1 \times \ldots \times G_m = R_0(G) \rhd R_1(G) \rhd \ldots \rhd R_k(G) = 1.$$

Here, $N_i = G \cap R_i(G)$ for each i.

We assume as inductive hypothesis that we already have the strong generating set of G relative to N_i, and we have a sifting procedure for elements of G with respect to this strong generating set. Furthermore, we assume inductively that computing this SGS and the sifting procedure can be carried out in the Mod_kL hierarchy of some level j. Here k is suitably chosen constant depending only on the orbit bound b.

We show using the above inductive assumption and Theorem 5.3 that the SGS of G rel N_{i+1} and can be computed in the $j+1$st level of the Mod_kL hierarchy, and the corresponding sifting procedure can be carried out in the $j+1$st level of the Mod_kL hierarchy.

Using the already computed the SGS of G relative to N_i and the corresponding sifting procedure, we can compute (in the jth level of the Mod_kL

hierarchy) a subset $S \subseteq G$ such that $\mathrm{NCL}_G(S) = N_i$ (Lemma 5.2). Now, using this set S, notice that $N_i = G \cap R_i(G)$ and $N_{i+1} = G \cap R_{i+1}(G)$ are groups to which we can apply Theorem 5.3, with $L = R_i(G)$ and $M = R_{i+1}(G)$, to find both the SGS of N_i rel N_{i+1} and be able to sift w.r.t. this SGS in $\mathrm{FL}^{\mathrm{Mod}_k\mathrm{L}}$. The union of the SGS of G rel N_i with the SGS of N_i rel N_{i+1} will give the SGS of G rel N_{i+1}. Clearly, the computation can be done in the $j+1$st level of the $\mathrm{Mod}_k\mathrm{L}$ hierarchy.

We next need to show how to sift w.r.t. this SGS of G rel N_{i+1}. Given an element $g \in G$ we first compute its sift g' w.r.t the SGS of G rel N_i. By induction hypothesis, this computation can be done in jth level of the $\mathrm{Mod}_k\mathrm{L}$ hierarchy. Next, since the element $g' \in G \cap N_i$, we can apply the $\mathrm{FL}^{\mathrm{Mod}_k\mathrm{L}}$ sift procedure of Theorem 5.3 w.r.t. to the SGS of N_i rel N_{i+1} to finally obtain the sift of g w.r.t. the SGS of G rel N_{i+1}. Again, the overall complexity lies in the $j+1$st level of the $\mathrm{Mod}_k\mathrm{L}$ hierarchy.

Continuing this for the l levels of the residue series, we obtain the required SGS for G. The computation can be carried out in the lth level of the $\mathrm{Mod}_k\mathrm{L}$ hierarchy. It is easy to see that the constant k can be chosen as the product of all distinct primes p such that N_i/N_{i+1} is $\mathbb{Z}_p$-semisimple. This completes the proof. ■

6 The Pointwise Stabilizer problem

The goal of this section is to prove Theorem 4.12 which would complete the upper bound proof for primitive-POINTSET$_b$, and hence for BCGI.

Let (G, Δ) be an instance of primitive-POINTSET$_b$. For each orbit Ω_i consider the projection of the residual tower given by Equation 1 to the orbit Ω_i resulting in the following series for G_i:

$$G_i = pr_i\left(R_0(G)\right) \rhd pr_i\left(R_1(G)\right) \rhd \ldots \rhd pr_i\left(R_k(G)\right) = 1.$$

Now, since each G_i is primitive, from Lemma 4.8, it follows that the smallest nontrivial group in the above series is $\mathrm{Soc}\left(G_i\right)$, for each i. Consequently, in the residue series $G = N_0 \rhd \ldots \rhd N_l$ of G, for each Ω_i there is some N_s such that $pr_i\left(N_s\right) = \mathrm{Soc}\left(G_i\right)$ and $pr_i\left(N_{s+1}\right) = \{1\}$. Given an index s, let X be the set of target orbit indices i such that $pr_i\left(N_s\right) = \mathrm{Soc}\left(G_i\right)$ and $pr_i\left(N_{s+1}\right) = \{1\}$.

Assume, as inductive hypothesis, that we have computed a generating set for the group H_s in the sequence of groups $G = H_0, H_1, \ldots, H_l$ claimed in Theorem 4.12 with the desired properties. Namely, H_s contains

$G_{\{\Delta\}}$ as subgroup and $|pr_i(H_s)| \leq |G_i|/2$ for all target orbit indices i such that $pr_i(N_s) = \{1\}$. We will find H_{s+1} as a subgroup of H_sN_s such that $|pr_i(H_s)| \leq |G_i|/2$ for all $i \in X$. Clearly, H_{s+1} will satisfy Theorem 4.12.

We first give a broad outline of the $\mathrm{FL}^{\mathrm{Mod}_t\mathrm{L}}$ algorithm for finding H_{s+1}. The actual details of the algorithm varies depends upon whether the socle is abelian or not. It will be taken up in the two subsections after the outline.

The algorithm will pick a *critical subset* Y of X and compute the following: For each generator x of H_s, it computes an element $x' \in G$ obtained as $xy \in H_sN_s$ for a suitably chosen $y \in N_s$ such that x' fixes $\Delta \cap \Omega_i$ for each $i \in Y$. Next, the algorithm computes the subgroup N of N_s which pointwise fixes $\Delta \cap \Omega_i$ for each $i \in Y$. Let $H = \langle \{x' \mid x \text{ generator of } H_s\} \rangle$. By construction, HN will pointwise fix $\Delta \cap \Omega_i$ for each $i \in Y$. Furthermore, the algorithm will choose Y suitably (which will be explained in the details) so that $|pr_i(HN)| \leq |pr_i(G)|/2$ for all $i \in X$. The subgroup H_{s+1} is defined to be HN.

Since the residue series for G has the property that for each i there is an s such that $pr_i(N_s)$ is the socle of G_i, it follows that $|pr_i(H_l)| \leq |G_i|/2$ for all the target orbits Ω_i.

When the socle is nonabelian

Suppose N_s/N_{s+1} is T-semisimple for a nonabelian simple group T. Based on the O'Nan Scott Theorem 4.10, we can partition X into $X_1 \cup X_2$, where for each $i \in X_1$, $\mathrm{Soc}(G_i)$ is of type (ii), and for each $i \in X_2$, $\mathrm{Soc}(G_i)$ is of type (iii). For $i \in X_1$, $\mathrm{Soc}(G_i)$ is the unique minimal normal subgroup of G_i. For $i \in X_2$, $\mathrm{Soc}(G_i)$ is of the form $K_1 \times K_2$, where the two *socle parts* K_1 and K_2 are the only two minimal normal subgroups of G_i.

Furthermore, if K is a socle for some $i \in X_1$, or K is a socle part for for some $i \in X_2$, then K is T-semisimple. More precisely, it is of the form $T_1 \times \ldots \times T_d$ for a constant d with each $T_j \cong T$, and G acts transitively by conjugation on $\{T_1, \ldots, T_d\}$.

In the following lemma we summarize observations from [Lu86] (that follow from the O'Nan Scott Theorem 4.10 and Scott's Lemma 4.11). This lemma identifies the group N_s projected on orbits in X using the socle parts of $\mathrm{Soc}(G_i)$ for $i \in X$. More precisely, if K and K' are two socle parts corresponding to any two indices $i, j \in X$, then N_s projected on $\Omega_i \cup \Omega_j$ either induces the group $\mathrm{Diag}(K \times K')$ or $K \times K'$. In the former case we say that the socle parts K and K' are *linked* in N_s.

Lemma 6.1. *Let $G = N_0 \rhd \ldots \rhd N_l$ be a residue series for the permutation group G, where (G, Δ) is an instance of primitive-POINTSET$_b$. Let X be*

the set of indices i such that $pr_i(N_s) = \mathrm{Soc}(G_i)$ and $pr_i(N_{s+1}) = \{1\}$ where N_s/N_{s+1} is T-semisimple for a nonabelian simple group T. For any pair of indices $i, j \in X$ exactly one of the following holds:

(a) N_s projected on $\Omega_i \cup \Omega_j$ is of the form $\mathrm{Soc}(G_i) \times \mathrm{Soc}(G_j)$.

(b) If $i, j \in X_1$ and N_s projected on $\Omega_i \cup \Omega_j$ is of the form $\mathrm{Diag}(K \times K')$, where $K = \mathrm{Soc}(G_i)$ and $K' = \mathrm{Soc}(G_j)$.

(c) If $i, j \in X_2$, and $\mathrm{Soc}(G_i) = K_1 \times K_2$ and $\mathrm{Soc}(G_j) = K_1' \times K_2'$ then N_s projected on $\Omega_i \cup \Omega_j$ is either of the form $\mathrm{Diag}(K_1 \times K_1') \times \mathrm{Diag}(K_2 \times K_2')$ or of the form $\mathrm{Diag}(K_1 \times K_1') \times K_2 \times K_2'$.

(d) If $i \in X_1$ and $j \in X_2$, $\mathrm{Soc}(G_i) = K$ and $\mathrm{Soc}(G_i) = K_1 \times K_2$, then N_s projected on $\Omega_i \cup \Omega_j$ is of the form $\mathrm{Diag}(K \times K_1) \times K_2$.

Furthermore, observe that if $K = T_1 \times \ldots \times T_d$ and $K' = T_1' \times \ldots \times T_e'$ are two linked socle parts, then $e = d$ and in $\mathrm{Diag}(K \times K')$, T_j is linked to $T'_{\pi(j)}$ for some permutation π of the indices.

In deterministic logspace we first construct an undirected graph $\mathcal{G}$ with vertex set $V = \{K \mid K \text{ is a socle part of } \mathrm{Soc}(G_i), i \in X\}$. The edge set E is partitioned into RED and BLUE edges defined as follows: RED $= \{(K, K') \mid K \text{ and } K' \text{ are linked}\}$ and BLUE $= \{(K, K') \mid \exists i \in X : \mathrm{Soc}(G_i) = K \times K'\}$.

Next, in deterministic logspace the algorithm will find the connected components in the red subgraph (V, RED). Notice that by Scott's Lemma 4.11 the red subgraph is transitive. Thus, the connected components are just cliques and can be easily determined in deterministic logspace.

Finding Y We partition the cliques of the red subgraph of $\mathcal{G}$ into two sets $\mathcal{T}_1$ and $\mathcal{T}_2$, where a red clique C is put in $\mathcal{T}_1$ if C contains an element $K = \mathrm{Soc}(G_i)$ for some $i \in X$. The remaining cliques are put in $\mathcal{T}_2$.

We will find the critical subset Y of X in two stages. In the first stage we pick Y_1 defined as follows: for every clique $C \in \mathcal{T}_1$ pick a $K = \mathrm{Soc}(G_i) \in C$ and include the index i in Y_1.

We now take the graph $\mathcal{G}$ and delete all the cliques in $\mathcal{T}_1$ (and the incident edges). Call the new graph $\mathcal{G}'$. In $\mathcal{G}'$, considering the red cliques as vertices, we compute the lexicographically first spanning forest of blue edges in $\mathrm{L}^{\mathrm{SL}} = \mathrm{L}$.[3]

[3] Alternatively, we can shrink the cliques in $\mathcal{G}'$ into vertices and find a spanning forest of blue edges.

Let E' be the blue edges in the spanning forest. Recall that each $e = (K, K') \in E'$ corresponds to the orbit Ω_i where $\mathrm{Soc}\,(G_i) = K \times K'$. The remaining part Y_2 of the critical subset Y consist of such indices i corresponding to edges in E'.

Thus, we have determined Y in $\mathrm{L}^{\mathrm{SL}} = \mathrm{L}$.

Finding N and H The subgroup N of N_s pointwise fixes $\Delta \cap \Omega_i$ for each $i \in Y$. I.e. $N = \{g \in N_s \mid \delta^g = \delta \ , \forall\ \delta \in \cup_{i \in Y}(\Delta \cap \Omega_i)\}$. Notice that, since N_{s+1} fixes $\Delta \cap \Omega_i$ for each $i \in X$, we have $N_s \geq N \geq N_{s+1}$.

We find N in two stages corresponding to Y_1 and Y_2.

Let $N' = \{g \in N_s \mid \delta^g = \delta,\ \forall\ \delta \in \cup_{i \in Y_1}(\Delta \cap \Omega_i)\}$.

Let the set S denote the SGS for N_s rel N_{s+1}. By the construction of the SGS described in Theorem 5.3 for the nonabelian case, it is easy to see that for every clique $C \in \mathcal{T}_1$, there is a subset S_C of the SGS S with the following property:
For a $K \in C$ let the corresponding orbit index be i (note that $i \in Y_1$). Either $\mathrm{Soc}\,(G_i) = K$ or $\mathrm{Soc}\,(G_i) = K \times K'$, and we have $pr_i\,(S_C) = K$ (note that K' has to be in a different clique). Furthermore, for each index $j \neq i$ we have $pr_j\,(S_C) = \{1\}$.

Now, from the elements of S_C for each $C \in \mathcal{T}_1$, we can compute the subgroup N' of N_s in deterministic logspace. Let $S'_C \subset S_C$ be those elements that fix $\Delta \cap \Omega_i$ for each $i \in Y_1$ corresponding to some $K \in C$. Then N' is generated by the union of the subsets S'_C of S_C for different $C \in \mathcal{T}_1$.

Furthermore, for each generator x of H_s, we can also compute an element $y \in S$ such that $xy = x'$ pointwise fixes $\cup_{i \in Y_1}(\Delta \cap \Omega_i)$. More specifically, y can be chosen by picking appropriate elements from different S_C, $C \in \mathcal{T}_1$ and multiplying them. Denote by H' the group generated by these x''s computed as above.

Next we explain how to handle the critical orbit indices in Y_2. Consider the spanning forest with vertex set $V' = V(\mathcal{G}')$ and edges E' (already defined, consisting of blue edges). Each tree in this forest can be considered as a rooted tree, with its root at the lexicographically least vertex. Thus, each edge in the forest is directed blue edge.

Consider any $x \in G$. We explain an $\mathrm{FL}^{\mathrm{SL}} = \mathrm{L}$ algorithm for computing an element $y \in N'$ such that xy fixes all the elements in $\cup_{i \in Y_2}(\Delta \cap \Omega_i)$. In order to do this we define an $\mathrm{FL}^{\mathrm{SL}} = \mathrm{L}$ subroutine that does the following:

Step 1. It takes as input an edge $\{u, v\}$ in the spanning forest (V', E'). Using an SL oracle it determines the the unique path $s = u_0, u_1, \ldots, u_{r-1} =$

$u, v = u_r$ in the forest from a root s, this gives the edge (u, v) an orientation (in the direction of the path from s).

Step 2. Let the actual socle parts determining the edges in this path be (K_i, L_i) for each edge (u_i, u_{i+1}). These can be identified easily in logspace. Now, suppose we have picked elements $y_0, \ldots, y_{i-1}$ of the SGS of N_s rel N_{s+1} for the first i edges of this path. Let the orbit corresponding to (K_i, L_i) be Ω_j and let $\{\delta\} = \Delta \cap \Omega_j$. If $\delta^x = \mu$ and $\mu^{y_{i-1}} = \nu$, then for the edge (u_i, u_{i+1}) pick the lexicographically least element y_i from the SGS of N_s rel N_{s+1} such that $\nu^{y_i} = \delta$. This is clearly possible since L_i is transitive and it is not linked to any socle part associated to an edge at distance smaller than $i + 1$.

Step 3. Output the elements y_i corresponding to the edges of G in increasing order of distance from their corresponding roots. This can also be done in $\mathrm{FL}^{\mathrm{SL}} = \mathrm{L}$.

Define y as a product of the y_i's computed in the order output by the above subroutine. It is easy to see by construction that $x' = xy$ will fix each point in $\cup_{i \in Y_2}(\Delta \cap \Omega_i)$.

Now, we can define the group H'' as the subgroup generated by these elements x' for each generator x of H'.

Likewise, let N'' be the subgroup of N' obtained by computing z' for each generator z of N' using the above subroutine. It is easy to see that N'' is the pointwise stabilizer of $\cup_{i \in Y}(\Delta \cap \Omega_i)$.

We define $H_{s+1} = H''N''$. The construction of H_{s+1} enforces that if K and K' are any two socle parts (vertices in V) that are connected by a path in $(V, \mathrm{RED} \cup \mathrm{BLUE})$ then in H_{s+1} the socle parts K and K' are diagonally linked. We first argue that in H_{s+1} for each $i \in X$, $|pr_i(H_{s+1})| \leq |G_i|/2$. This is already true for each $i \in Y$ by construction. Let $i \in X \setminus Y$.

If $\mathrm{Soc}(G_i)$ is of type (ii), then the corresponding socle part K is in a red clique. Furthermore, since the red clique has a representative L in Y on which H_{s+1} is cut down, due to the linking of K and L, H_{s+1} is cut down in K (and hence G_i) as well.

If $\mathrm{Soc}(G_i)$ is of type (iii), let $\mathrm{Soc}(G_i) = K_1 \times K_2$. Now, $pr_i(H_{s+1}) = \mathrm{Diag}(K_1 \times K_2)$, implying that H_{s+1} is cut down on this orbit as well.

Let $\Sigma = \cup_{i \in Y}(\Delta \cap \Omega_i)$. By construction we have $H_s \leq H_{s+1}N_s$. Now, inductively assuming that $G_{\{\Delta\}} \leq H_s$, notice that

$$G_{\{\Delta\}} \leq (H_s)_{\{\Sigma\}} \leq (H_{s+1}N_s)_{\{\Sigma\}} = H_{s+1}N'' = H_{s+1}.$$

This completes the proof for the case when the socle is nonabelian.

When the socle is abelian

We now turn to the case when $\mathrm{Soc}\,(G_i)$ is abelian for each $i \in X$, where X is the set of orbit indices for which $pr_i\,(N_s)$ is abelian. Assume that the group H_s is already computed.

Finding Y Suppose N_s/N_{s+1} is T-semisimple. Then T is $\mathbb{Z}_p$ for some prime p. By the O'Nan Scott Theorem 4.10, for each i $\mathrm{Soc}\,(G_i)$ is a subgroup of $\mathbb{Z}_p^{c_i}$, where the number of copies of $\mathbb{Z}_p$ is a constant c_i (that depends on b). It follows that N_s projected on the set of orbits Ω_i for $i \in X$ is isomorphic to a subgroup of $\mathbb{Z}_p^r$ where $r = \sum_{i \in X} c_i = O(m)$. Thus, N_s projected on the set of orbits Ω_i for $i \in X$ is actually a subspace U of $\mathbb{Z}_p^r$. Here, in $\mathbb{Z}_p^r$ we assume that the coordinates corresponding each orbit $i \in X$ are adjacent and appear in the increasing order of $i \in X$. Furthermore, we can compute a basis for this projected subspace U from the generators of N_s in deterministic logspace. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, v_q$ be the computed basis. In $\mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$ we can compute a special basis for the vector space U as follows: for each index j, $1 \leq j \leq r$ form a column vector $y^{(j)} = (0, \ldots, 0, 1, y_{j+1}, \ldots, y_r)^T$, where the 1 occurs in the jth position, the first $j-1$ positions have 0, and $y_{j+1}, \ldots, y_r$ are indeterminates. Let A denote the matrix with columns as $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q$. In $\mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$ the feasibility of $Ax = \mathbf{y}^{(j)}$ can be tested, and if feasible then a solution $\mathbf{y}^{(j)}$ to $y^{(j)}$ can be computed. These vectors $\mathbf{y}^{(j)}$ clearly form a basis B for U.

Define the subset of orbit indices $Y \subset X$ as follows: include index i in Y if for some index j corresponding to Ω_i, there is a solution $y(j) \in B$.

Finding N and H Now, projecting the vectors in B only on the orbits of Y, we again apply an $\mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$ computation to obtain a new set of vectors B' such that B' spans U projected on the orbits of Y, and for each orbit index $i \in Y$ there is a vector $z_i \in B'$ such that z_i is nontrivial on all orbits of Y except Ω_i. As mentioned before, notice here that the algorithm can keep track of the corresponding entire permutation in G while working with a vector at each stage. Thus, we also have a $\pi_i \in N_s$ which coincides with z_i on the orbits Y.

Now, let x be a generator of H_s. By the above construction we can obtain an element $y \in N_s$ as a product of the form $\prod_{i \in Y} \pi_i^{k_i}$ such that $x' = xy \in H_s N_s$ fixes $\Delta \cap \Omega_i$ for each $i \in Y$.

Let H be the subgroup generated by the above elements x'.

Next, notice that the subgroup $N \leq N_s$ consisting of elements that fix $\cup_{i \in Y} \Delta \cap \Omega_i$ is trivial on all the orbits in Y (by O'Nan Scott Theorem).

Furthermore, the definition of Y forces that N is trivial on all Ω_i, $i \in X$.

Hence, notice that $N_{s+1} \lhd N \lhd N_s$. Thus, N_s/N is also $\mathbb{Z}_p$-semisimple. To find a generating set for N we will apply the algorithm of sifting and normal closure from Theorem 5.3. To sift we proceeds as follows: for each generator x of N_s, find an element $y \in N_s$ as a product of the form $\prod_{i \in Y} \pi_i^{k_i}$ such that $x' = xy$ fixes $\Delta \cap \Omega_i$ for each $i \in Y$. Let S' be this set of elements x', which is the sift of x in N_s rel N. It remains to compute $\mathrm{NCL}_{N_s}(S')$. This can be easily done in $\mathrm{FL}^{\mathrm{Mod}_p\mathrm{L}}$ by applying the algorithm described in the abelian case of Theorem 5.3 to the set S'.

Now, consider the group H_{s+1} defined as HN. Since for each $i \in X$, $pr_i(H \cap N_s)$ is trivial, by Lemma 4.9 it follows that $|pr_i(H_{s+1})| \leq |G_i|/2$ for each orbit $i \in X$.

Notice that by construction of H_{s+1} we have $H_s \leq H_{s+1}N_s$. Let $\Sigma = \cup_{i \in Y}(\Delta \cap \Omega_i)$. Now, inductively assuming that $G_{\{\Delta\}} \leq H_s$, notice that

$$G_{\{\Delta\}} \leq (H_s)_{\{\Sigma\}} \leq (H_{s+1}N_s)_{\{\Sigma\}} = H_{s+1}N = HNN = HN = H_{s+1}.$$

This completes the proof of the abelian case, and hence the proof of Theorem 4.12.

Acknowledgments. We thank Jacobo Torán for discussions with him on applying his result [Tor04, Lemma 4.7] to derive that BCGI is hard for the $\mathrm{Mod}_k\mathrm{L}$ hierarchy. The first and second authors are grateful to Johannes Köbler for hosting their visits at Humboldt Universität, Berlin under the DST-DAAD project, and for many useful discussions on BCGI and related questions.

References

[ABO99] Eric Allender, Robert Beals, and Mitsunori Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8:99-126, 1999.

[AO96] E. Allender and M. Ogihara. Relationships among PL, #L and the determinant. *RAIRO Theoretical Informatics and Applications,* 30(1):1–21, 1996.

[ARZ99] E. Allender, K. Reinhardt, and S. Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164-181, 1999.

[Al04] E. Allender. Arithmetic Circuits and Counting Complexity Classes. To appear in *Quaderni di Matematica* series, Edited by Jan Krajicek, 2004.

[Bab79] L. Babai Monte Carlo algorithms in graph isomorphism testing. *Universitat de Montreal Tech. Report D.M.S*, 79-10, 1979.

[BDHM92] G. Buntrock, C. Damm, U. Hertrampf, C. Meinel. Structure and Importance of Logspace-MOD Classes. *Mathematical Systems Theory,* 25(3): 223-237 (1992).

[FHL80] M. L. Furst, J. E. Hopcroft, E. M. Luks. Polynomial-Time Algorithms for Permutation Groups. In *Proceedings of Foundations of Computer Science,* IEEE Computer Society, 36-41, 1980.

[Ha] M. Hall. *The Theory of Groups.* Macmillan, New York, 1959.

[JKMT04] B. Jenner, J. Köbler, P. McKenzie, J. Torán. Completeness results for graph isomorphism. *J. Comput. Syst. Sciences,* 66(3): 549-566, 2003.

[Lu86] E. M. Luks. Parallel algorithms for permutation groups and graph isomorphism. In *Proceedings of the IEEE Foundations of Computer Science*, IEEE Computer Society, 292-302, 1986.

[Luk93] E. M. Luks. Permutation groups and polynomial time computations. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.

[LM88] E. M. Luks and P. McKenzie. Parallel algorithms for solvable permutation groups. *Journal of Computer and System Sciences,* 37, 1988, 39-62.

[MC87] P. McKenzie and S. C. Cook. The parallel complexity of abelian permutation group problems. *Siam Journal of Computing,* 16:880-909, 1987.

[Re04] O. Reingold. Undirected graph connectivity is in logspace. *ECCC Technical Report,* TR94-04, Nov. 2004.

[RST84] W. L. Ruzzo, J. Simon, and M. Tompa. Space-bounded hierarchies and probabilistic computation. *Journal of Computer and System Sciences,* 28:216-230, 1984.

[Tod91] S. Toda. Counting problems computationally equivalent to computing the determinant. Technical Report CSIM 91-07, Department of Computer Science, University of Electro-Communications, Tokyo, Japan, May 1991.

[Tor04] J. Torán. On the hardness of Graph Isomorphism. *SIAM Journal of Computing,* 33(5): 1093-1108, 2004.

[Vin91] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Structure in Complexity Theory Conference*, pages 270-284, 1991.

[Wie64] Helmut Wielandt. *Finite Permutation Groups.* Academic Press, New York, 1964.