The Complexity of NP Problems over Finite Groups

V. Arvind and N. V. Vinodchandran
Computer Science Group
Institute of Mathematical Sciences, C.I.T Campus
Madras 600113, India
email: arvind,vinod@imsc.ernet.in

1 Introduction

The Graph Isomorphism problem (GI, of deciding whether there is an isomorphism between two labeled graphs) is one of the very well-studied algorithmic problems. While, to date, there is no known polynomial-time algorithm for graph isomorphism, it is surprising that it is unlikely to be NP-complete. In fact, there are strong evidences that the graph isomorphism problem is not NP-complete. Mathon's result that the problem of counting the number of isomorphisms between two labeled graphs is Turing reducible to GI, was the first indication that GI is unlikely to be NP-complete. In [BHZ87] it was shown that the complement of the decision problem GI is in AM implying that GI is in NP $\cap$ co-AM. Consequently, GI cannot be NP-complete unless the polynomial hierarchy collapses to Σ_2^p [BHZ87, Sch88]. More recently, it was shown in [KST92] that GI is *low* for PP. In other words, GI is powerless as oracle for PP computations (where PP is the language class corresponding to #P).

Similar results also hold for several group-theoretic problems. In [KST92] it is shown for permutation groups that, the Group Intersection problem, Group Factorization, Coset Intersection and Double Coset Memebership (formal definitions are given later) are all low for PP. In [BaMo88] it is shown that Group Intersection and the Coset Intersection problems for permutation groups are in NP $\cap$ co-AM.

The underlying group-theoretic structure of these problems plays a crucial role in the proofs of the above-mentioned complexity results. For example, in the case of Graph Isomorphism the proofs of the results rely crucially on the fact that the set of automorphisms of a graph on n vertices is a subgroup of S_n, the symmetric group on n elements. Furthermore, if G_1 and G_2 are two isomorphic graphs and $Aut(G_1)$ denotes the automorphism group of G_1 then the set of all isomorphisms from G_1 to G_2 is a right coset $Aut(G_1)\psi$ of $Aut(G_1)$, where ψ is an isomorphism from G_1 to G_2. Similarly, in the case of other group theoretic problems, one can easily observe the role of their group theoretc structure in deriving the above-mentioned upper bounds.

This work is motivated by the intriguing complexity of Graph Isomorphism problem and the above-mentioned group theoretic problems. We would like to know the complexity of those problems which enjoy similar group theoretic structure, in a unified, generalised framework. More precisely, we study the complexity of an NP set A which has the property that for each instance x of A, the set of solutions for x has a group structure imposed on it. To make this notion precise, we define the class of problems $\mathcal{L}_{rc}$ consisting of problems in NP whose solution sets form right coset of some balck-box group. Then we show results which essentially indicate that, these NP problems are unlikely to be NP-complete. Firstly, we show that the counting problems corresponding to these NP sets are in $\mathrm{FP}^{\mathrm{AM}}$, and hence unlikely to be #P complete unless PH collapses to $\mathrm{FP}^{\mathrm{AM}}$. This indicates that these NP problems are unlikely to be NP-complete. In the next section, we look at the complexity of NP problems whose solution sets form a vector space, and show that these problems are in the class SPP. More importantly, in the last section, we show that, if we restrict the blck-box group to be *abelian* ($\mathcal{L}_{rc}^{abel}$) then these NP problems are low for the class PP. This result, though highly technical, gives lowness results for various group theoretic problems as corollary. For proving the result about NP sets over abelian groups, we use the fundamental theorem of abelian groups.

2 Notations and Definitions

We fix the finite alphabet $\Sigma = \{0, 1\}$. For any oracle A, the language L in $\mathrm{NP}(A)$ if there is a relation $R \in \mathrm{P}(A)$ and a polynomial p such that $x \in L$ iff $\exists y \in \Sigma^{p(|x|)}$ and $\langle x, y\rangle \in R(A)$. Corresponding to the relation R let $Sol_R(x)$ denote the set $\{y \in \Sigma^{p(|x|)} \mid \langle x, y\rangle \in R\}$, for each $x \in \Sigma^*$.

Let $\mathbf{Z}$ denotes the set of integers. A function $f : \Sigma^* \to \mathbf{Z}$ is *gap-definable* if there is a polynomial-time nondeterministic (in short, NP) machine M such that, for each $x \in \Sigma^*$, $f(x)$ is the difference between the number of accepting paths (denoted by $acc_M(x)$) and the number of rejecting paths (denoted by $rej_M(x)$) of M on input x. Let GapP [FFK91] denote the class of gap-definable functions. For each NP machine M let gap_M denote the GapP function defined by it. The corresponding language class PP is defined as follows: A language L is in PP if there is a GapP function f such that: $x \in L$ iff $f(x) > 0$.

The counting classes of interest in this paper are UP, FewP, SPP and LWPP defined using GapP functions [FFK91]. A language L is in UP if there is an NP machine M accepting L such that M has at most one accepting path on any input. A language L is in FewP if there is polynomial p and an NP machine M accepting L such that M has at most $p(|x|)$ accepting for any input. A language L is in SPP if there is an $f \in$ GapP such that: $x \in L$ implies that $f(x) = 1$, and $x \notin L$ implies that $f(x) = 0$. A language L is in LWPP if there are functions $f \in$ GapP and $h \in$ FP such that: $x \in L$ implies that $f(x) = h(0^{|x|})$, and $x \notin L$ implies that $f(x) = 0$. It is known that UP $\subseteq$ SPP $\subseteq$ LWPP.

For an oracle $A \in \Sigma^*$ let $\mathrm{P}(A), \mathrm{NP}(A)$ and $\mathrm{PP}(A)$ denote the classes of languages obtained by the natural relativization w.r.t. oracle A. $A \in \Sigma^*$ is said to be *low* for PP if $\mathrm{PP}(A) = \mathrm{PP}$. In [FFK91] it is shown that every language in LWPP is low for PP.

Another complexity class which is of interest to us is the class AM introduced in [Ba85]. We refer the reader to [Ba85] for definition of this class.

2.1 Group-theoretic preliminaries

In this subsection, we give some basic group-theoretic definitions and formalize some notation. The reader can refer standard textbooks [Bu, Hall] for details.

Definition 2.1 [Bu, Hall] A *group* is a nonempty set G endowed with a binary operation $*$ such that G is closed under $*$. The operation $*$ is associative. There is an element $e \in G$, called the *identity* of G, such that $x * e = e * x = x$, for all $x \in G$. For every $x \in G$ there exists a unique $x^{-1} \in G$, called the *inverse* of x, such that $x * x^{-1} = x^{-1} * x = e$.

When there is no ambiguity we do not explicitly specify the group operation and denote the composition $x * y$ by xy.

A group G is a *finite group* if the cardinality of the set G (denoted by $|G|$) is finite. In the sequel G stands for a finite group. The *order* of G is defined as $|G|$. For an element $g \in G$, the *order* of g (denoted $o(g)$) is the smallest positive integer such that $g^{o(g)} = e$, where e is the identity of G. A subset H of G is called a *subgroup* of G (denoted $H < G$) if H is a group under the group operation of G.

A group G is *abelian* if $\forall g_1, g_2, \in G : g_1 g_2 = g_2 g_1$. Let S be a subset of a group G. The smallest subgroup of G containing S is called the group *generated by* S and is denoted $\langle S \rangle$. A subset S of G is a generator set for G if $G = \langle S \rangle$. Observe that, if S is a set of generators for G then G is abelian iff $\forall g_1, g_2, \in S : g_1 g_2 = g_2 g_1$.

Let H be a subgroup of a group G. For $g \in G$ the set $\{hg \mid h \in H\}$ denoted by Hg is a *right coset* of H in G. Similarly, the set $gH = \{gh \mid h \in H\}$ is a *left coset* of H in G.

A subgroup H of G is a *normal* subgroup of G if for all $g \in G$ it holds that $Hg = gH$. As a consequence it turns out that the set $G/H = \{Hg \mid g \in G\}$ is a group (the *quotient group* induced by H) under the binary operation $\cdot$ defined by $Hx \cdot Hy = Hxy$.

Let p be a prime. A p-group is a finite group whose order is a power of p. Let G be finite groups such that $|G| = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$. A fundamental theorem of group theory is that for each i there is a subgroup of G of order $p_i^{e_i}$. A subgroup of G of order $p_i^{e_i}$ is referred to as a p_i-*Sylow subgroup* of G.

Let $(X, *)$ and $(Y, .)$ be two groups. The *direct product* of the groups X and Y is defined as the group $(X \times Y, \circ)$, where $X \times Y$ is the cartesian product of sets X and Y, and for $(x_1, y_1), (x_2, y_2) \in X \times Y$ their $\circ$ composition is defined as $(x_1, y_1) \circ (x_2, y_2) = (x_1 * x_2, y_1 . y_2)$.

Next, we make precise our definition of the class of NP problems, the solution sets of whose instances form right cosets of finite groups. We require that the group operations to be computable in polynomial time. Let PROD, INV and ID be class of partial functions such that:
PROD $= \{prod : \Sigma^* \times \Sigma^* \to \Sigma^* \cup \{\perp\} | prod \in \text{FP}\}$.
INV $= \{inv : \Sigma^* \to \Sigma^* \cup \{\perp\} | inv \in \text{FP}\}$.
ID $= \{id : \Sigma^* \to \{true, false\} | id \in \text{FP}\}$.

Definition 2.2 Let $prod \in$ PROD, $inv \in$ INV and $id \in$ ID.

1. A subset $X \subseteq \Sigma^*$ is said to be a group defined by $(prod, inv, id)$ if (1). $\forall x, y, z \in X, prod(x, prod(y, z)) = prod(prod(x, y), z)$ (Associativity). (2). Forall $x \in X$, $id(prod(x, inv(x)) = id(prod(inv(x), x)) = true$ (Existance of inverse) and (3). There exists a unique $i \in X$ such that $id(i) = true$ and $\forall x \in X, prod(x, i) = prod(i, x) = x$ (Existance of identity). We denote the group defined as $(X, prod, inv, id)$.

2. A subset $X \subseteq \Sigma^*$ is said to be an *abelian* group defined by $(prod, inv, id)$ if X is a group defined by $(prod, inv, id)$ and $\forall x, x' \in X, prod(x, x') = prod(x', x)$.

3. A subset $Y \subseteq \Sigma^*$ is said to be a *right coset* defined by $(prod, inv, id)$ if there exists $X \subseteq \Sigma^*$ such that X is a group defined by $(prod, inv, id)$, $Y \subseteq X$ and $\exists x \in X$ such that $Y = \{prod(x, x') | x' \in X\}$.

It is to be noted here that the groups we defined by polynomial time operations fit into the framework of *Black-box groups* defined by Babai and Szemeredi in [BaSz85]. Their definition is framed in such a way that it can also incorporate factor groups. In black-box groups of [BaSz85], group operations are done by an oracle. So any upper bound (worse than polynomial time) on black-box group problems will give an upperbound on the groups defined by polynomial time operations.

We now see some well known examples of classes of groups which can be seen as groups defined by polynomial time operations. Let S_n denote the permutation group on n elements. Then, elements of S_n can be coded in Σ^b where $b = O(n \log n)$ for all n. It is easy to see that we can define functions $prod_{\mathrm{Sym}} \in$ PROD, $inv_{\mathrm{Sym}} \in$ INV and $id_{\mathrm{Sym}} \in$ ID such that these are the group opeartions in S_n. As another example let $GL_n(q)$ denote the group of all $n \times n$ invertible matrices over the finite field F_q of size q. It is easy to see that this also fits into our framework of groups defined by polynomial time operations.

We are interested in studying NP sets whose solution sets has an algebraic structure imposed on them. Motivating from the well known problems, we now define these NP sets below.

Definition 2.3 1. A set $L \subseteq \Sigma^*$ is said to be an NP set defined over right cosets of a group, if $L \in$ NP witnessed by polynomial time relation R and there exists $prod_L \in$ PROD, $inv_L \in$ INV and $id_L \in$ ID such that for all $x \in L, Sol_R(x)$ is a right coset defined by $(prod_L, inv_L, id_L)$. We say that L is an NP set defined over right cosets of a group witnessed by $(prod_L, inv_L, id_L)$. If $(prod_L, inv_L, id_L)$ defines an abelian group, then we say that L is said to be an NP set defined over right cosets of an ablian group.

2. A set $L \subseteq \Sigma^*$ is said to be an NP set defined over a *non-trivial* subgroup if there is a polynomial time relation R, $prod_L \in$ PROD, $inv_L \in$ INV and $id_L \in$ ID such that $\forall x \in \Sigma^*, Sol_R(x)$ is a group defined by $(prod_L, inv_L, id_L)$ and for all $x \in L$, $Sol_R(x)$ contains more than one element.

We call NP sets defined over right cosets of a group witnessed by $(prod_{\mathrm{Sym}}, inv_{\mathrm{Sym}}, id_{\mathrm{Sym}})$ (as in the example above) NP sets defined over right cosets SYM.

It is easy to see from the definition that GI and problems Group Factorization, Coset Intersection and Double Coset membership for permutation groups are NP sets defined over right cosets SYM. In [KST92], it is shown that these problems are in the class LWPP and hence low for PP. Now, using techniques similar to the techniques used in [KST92], we can prove the following theorem.

Theorem 2.4 *If a set A is an NP sets defined over right cosets SYM then it is in* LWPP *and hence low for* PP.

We omit the proof of this theorem since it is a straight adaptation of their proof. Motivated from these examples, in the remaining sections we explore the complexity of NP problems with an arbitrary group theoretic structure on their solution set.

3 NP problems over finite groups: counting

In this section we prove upper bounds on the complexity of the exact counting problem corresponding to NP sets defined over right cosets of groups. More precisely, we show that, for NP sets over right cosets of (*prod*, *inv*, *id*) for any *prod* $\in$ PROD, *inv* $\in$, INV and *id* $\in$ ID, counting the number of solutions can be done in $\mathrm{FP}^{\mathrm{AM}}$ and particularly, for NP sets over right cosets of SYM, counting the number of solutions can be done in $\mathrm{FP}^{\mathrm{NP}}$ with parallel queries.

As an intermediate result, we prove upperbounds on the following generalization of a problem posed by Hoffman [Ho82]: Given a polynomial-time membership test for a subgroup H of a group G (presented by a set of generators), to compute a set of generators for H. An interesting aspect of this problem is that it is a kind of converse of the usual problem of membership testing in a group presented by a generator set.

Before we prove these results, we state some results from [Ba92].

Theorem 3.1 [Ba92] *Let $\mathcal{B} = \{B_m\}_{m\geq 1}$ be a group family. Then the following sets are in* AM.

1. *Non-Membership* $\stackrel{\text{def}}{=} \{(0^m, S, g) \mid S \subseteq B_m, g \in B_m$ *and* $g \notin \langle S \rangle\}$.

2. *Order Verification* $\stackrel{\text{def}}{=} \{(0^m, S, n) \mid S \subseteq B_m$ *and* $|\langle S \rangle| = n\}$.

The following theorem gives an upperbound on computing a generator set for a group for which membership testing can be done in polynomial time.

Theorem 3.2 *Let $\mathcal{B} = \{B_m\}_{\geq m}$ be a group family. Let $H < B_m$. Then if a polynomial time membership test is given for H, there is an* $\mathrm{FP}^{\mathrm{AM}}$ *algorithm for computing a generator set for H. If $\mathcal{B} =$* Sym*, then there is an* $\mathrm{FP}^{\mathrm{NP}}$ *algorithm for computing a generator set for H.*

Proof. Let $\mathcal{B} = \{B_m\}_{\geq m}$ be a group family. Let A be the be defined as $A = \{\langle 0^m, S, g' \rangle \mid \exists y \in \Sigma^{p(m)-|g'|}; S \subseteq H; g'y \in H$ and $g'y \notin \langle S \rangle\}$ where p is a polynomial. First, we have the following easy claim.

Claim 3.2.1 $A \in$ AM. *In paritcular, if the group family $\mathcal{B} =$* Sym*, $A \in NP$.*

Proof First note that $A \in \exists$AM. This follows from part 1 of Theorem 3.1 and the fact that membership in H can be done in polynomial time. Since $\exists$AM $=$ AM, the claim follows for general black-box groups. In the case when the group family is Sym, the calim follows from the fact that, membership testing for permutation groups can be done in polynomial time [FHL79]. ■

We next give a description of a deterministic algorithm which uses A to compute a generator set of H. Since, in general $A \in$ AM and in the case when the group family is Sym $A \in$ NP the theorem follows.

```
GEN-COMPUTE(0^m)
  S ← {e}
  while ⟨0^m, S, ϵ⟩ ∈ A
        do g ← ϵ
```

4 **while** $\langle 0^m, S, g\rangle \in A$
5 **do if** $\langle 0^m, S, g1\rangle \in A$
6 **then** $g \leftarrow g1$
7 **else** $g \leftarrow g0$
8 **end-while** (*$Computes\ g \in H - \langle S\rangle$ by prefix search*)
9 $S \leftarrow S \cup \{g\}$
10 **end-while** (* *Computes a generator set for* G_x*)
11 **return** S

We shall see that the above algorithm computes a generator set for H in polynomial time. Note that the algorithm returns a set S when the **while**-loop condition of *line*-2 is violated. From the definition of the the language A, this happens when $\langle S\rangle = H$. This shows that the algorithm computes a generator set for H. Note that after each iteration of **while**-loop of *line*-2, an element $g \in G$ but not in $\langle S\rangle$ is contructed and added to the set S befor the next iteration. From Lagrange's theorem, it follows that $|\langle S \cup g\rangle| \geq 2|\langle S\rangle|$ if $g \notin S$. So it follows that the the **while**-loop of *line*-2 will be iterated at most $p(m)$ times where $p(m)$ is the length of strings encoding the elements of B_m. ∎

As an interesting corollary to the above theorem, we prove upperbound on the complexity of the following problem: Given two black-box groups A and B presented by a generator set; compute a generator set for the group $A \cap B$.

Corollary 3.3 *Let $\mathcal{B} = \{B_m\}_{\geq m}$ be a group family. Let $A, B \subseteq B_m$, then a generator set for $\langle A\rangle \cap \langle B\rangle$ can be computed in* $\mathrm{FP}^{\mathrm{AM}}$.

Proof Sketch. Define a language L as follows: $A = \{\langle 0^m, S, A, B, g'\rangle \mid \exists y \in \Sigma^{p(m)-|g'|};\ S \subseteq \langle A\rangle \cap \langle B\rangle;\ g'y \in \langle A\rangle \cap \langle B\rangle$ and $g'y \notin \langle S\rangle\}$. (Notice that the only difference between the language A in the above theorem and L is that the H in the definition of A is replaced by $\langle A\rangle \cap \langle B\rangle$ in L).

In [BaSz85] it is shown that the problem of membership testing in a black-box group presented by a generator set is in NP. So from part 1 of Theorem 3.1, it follows that Membership Testing is in NP $\cap$ co-AM. From [Sch88] we know that NP $\cap$ co-AM is low for AM. So it follows that the language $L \in AM$. So in the algorithm GEN-COMPUTE, if we replace the language A by L, we will get an algorithm for computing a generator set for $\langle A\rangle \cap \langle B\rangle$. ∎

We use the algoritm GEN-COMPUTE to compute the #P function corresponding to a language in $\mathcal{L}_{rc}(\mathcal{B})$ for any group family $\mathcal{B}$. We do it in two parts. In the first part (which is a corollary to the above theorem) we show that for any input in the language, a generator set for the group whose right coset is the solution set, can be computed in $\mathrm{FP}^{\mathrm{AM}}$. In the next part, we use this set of generators to compute the corresponding #P function.

Corollary 3.4 *Let $\mathcal{B}$ be any group family and $L \in \mathcal{L}_{rc}(\mathcal{B})$, w.r.t. the polynomial-time computable relation R. For an input $x \in L$, let $G_x < B_m$ denotes the group whose right coset is $Sol_R(x)$. Then a set of generators for G_x can be computed in* $\mathrm{FP}^{\mathrm{AM}}$.

Proof Sketch. The algorithm is very similar to the algorithm GEN-COMPUTE. On an input x, using prefix search, first compute a solution $y \in Sol_R(x)$. This can be done in $\mathrm{FP}^{\mathrm{NP}}$. Now, membership in G_x can be done in polynomial time, since to check whether an element g is in G_x, it is enough to check whether $y^{-1}g \in Sol_R(x)$, which can be done in polynomial time. So it follows from Theorem 3.2 a generator set can be computed in $\mathrm{FP}^{\mathrm{AM}}$. ∎

Theorem 3.5 *Let $\mathcal{B}$ be any group family and $L \in \mathcal{L}_{rc}(\mathcal{B})$, w.r.t. the polynomial-time computable relation R. Then, $\forall x, |Sol_R(x)|$ can be computed in* $\mathrm{FP}^{\mathrm{AM}}$.

Proof. Let $L \in \mathcal{L}_{rc}(\mathcal{B})$. Given $x \in \Sigma^*$ as input, we describe an $\mathrm{FP}^{\mathrm{AM}}$ algorithm for computing the #P function $|Sol_R(x)|$. Let p the polynomial such that $Sol_R(x) \subseteq \Sigma^{p(|x|)}$. For input x, let $G_x < B_m$ denotes the group whose right coset is $Sol_R(x)$. Note that $|G_x| = |Sol_R(x)|$. Let $B' = \{\langle n, S\rangle \mid \text{n is the order of the group } \langle S\rangle\}$. The corresponding prefix language B is defined as: $B = \{\langle m, S\rangle \mid \exists n \in \Sigma^{\leq(p(|x|)-|m|)} \text{mn is the order of the group } \langle S\rangle\}$. From part 2 of Theorem 3.1 and the fact that $\exists\mathrm{AM} = \mathrm{AM}$, it follows that $B \in AM$.

Now we give an algorithm EXACT-COUNT, which on input x, first simulates the $\mathrm{FP}^{\mathrm{AM}}$ of Theorem 3.4 to first compute a generator set for G_x and then using the language B as oracle, computes $|G_x|$ in polynomial time.

EXACT-COUNT(x)

```
Simulating the algorithm in Theorem 3.4 compute a generator set for G_x in FP^AM
m ← ε
while ⟨m, S⟩ ∈ B
    do if ⟨m1, S⟩ ∈ B
          then m ← m1
          else  m ← m0
 end-while (* Computes |G_x| by prefix search*)
return m
```

From the description of the algorithm, it is clear that it computes the order of G_x in polynomial time using a language in AM.

∎

Now, we have the following corollary, which essentially indicates that the counting the number of solutions of an NP set defined over an arbitrary group family cannot be #P-complete. Corollary follows from Toda's theorem[To91] that $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$.

Corollary 3.6 *Let $\mathcal{B}$ be any group family and $L \in \mathcal{L}_{rc}(\mathcal{B})$, w.r.t. the polynomial-time computable relation R. Then, if the function $f(x) = |Sol_R(x)|$ is #P-complete then* $\mathrm{PH} \subseteq \mathrm{P}^{\mathrm{AM}}$.

Next, we observe that in the special case where the group family is Sym, the Theorem 3.5 can be improved to get an $\mathrm{FP}_{||}^{\mathrm{NP}}$ algorithm to compute the #P function. We omit the proof which is essentially the same as that of Mathon's result [Ma79] that the number of isomorphisms between two graphs can be computed in polynomial time with parallel queries to GI.

Theorem 3.7 *Let $L \in \mathcal{L}_{rc}(\mathrm{Sym})$. Let R be the polynomial time relation for $L \in$ NP. Then $\forall x$, $\mid Sol_R(x) \mid$ can be computed in $\mathrm{FP}_{||}^{\mathrm{NP}}$.*

4 NP Problems over Vector Spaces

In this section, we restrict to the case when the black-box groups are subspaces of a finite dimentional vector space and prove an upper bound on their complexity. Let F_q be the field of cardinality q. We show that the class $\mathcal{L}_{VS}(q)$ (defined below), is in the complexity class SPP. (Notice that we need to have a slight modification in the definition in the case when the solution set form a subspace, since the 0 vector will be there in all subspaces and to check whether a string is in the language, it is enough to check whether the string encoding the 0 vector is inside the solution set.)

We formally define the class of languages in NP, whose solution sets have a vector space structure imposed on it. Before that we introduce some notations. Let F_q be the field of cardinality q. Let F_q^* denotes the set of all invertible elements of F_q. F_q^n denotes the vector space which is the direct sum of F_q, n times. For any vector $\mathbf{y} \in F_q^n$, y_i denotes the i^{th} projection of vector $\mathbf{y} \in F_q^n$. Let 0 denotes the additive identy of F_q and $\mathbf{0}$ denotes the zero vector of the vector space. Let $V < F_q^n$ denotes that V is a subspace of F_q^n and V_α denotes the set $\{\mathbf{y} \in V | y_i = \alpha\}$.

Definition 4.1 Let F_q be the field of cardinality q. A language $A \subseteq \Sigma^*$ is said to be in $\mathcal{L}_{VS}(q)$, if there is a polynomial p and a polynomial time computable relation R, such that for every $x \in \Sigma^*$, $Sol_R(x)$ is a subspace of the vector space $F_q^{p(|x|)}$ and $A = \{x | Sol_R(x) \neq \{\mathbf{0}\}\}$, where $\mathbf{0}$ is the zero vector in $F_q^{p(|x|)}$.

We next state a proposition from linear algebra the proof of which can be seen in [HK]

Proposition 4.2 *Let $V < F_q^n$. Then for any $\alpha \in F_q$, if $V_\alpha \neq \phi$ then $|V_\alpha| = |V_0|$.*

Theorem 4.3 *Let F_q denotes the finite field of cardinality q. Then the class $\mathcal{L}_{VS}(q) \subseteq$ SPP.*

Proof. Let L be a language in $\mathcal{L}_{VS}(q)$ via the polynomial time relation R and polynomial p. Define a language L' as follows. $L' = \{\langle x, i, \alpha\rangle | x \in L; 1 \leq i \leq p(|x|)$ and $\alpha \in F_q\}$. The following non-deteministic machine M' witnesses $L' \in NP$.

MACHINE $M'(x, i, \alpha)$
1 **Guess** $y \in \Sigma^{p(|x|)}$
2 Verify the following in polynomial time
3 (a) y encodes a nonzero vector $\mathbf{y} \in F_q^{p(|x|)}$
4 (b) $\mathbf{y} \in Sol_R(x)$
5 (c) $y_i = \alpha$

Now we shall define a function $f \in$ GapP such that if $x \in L$, then $f(x) = 0$ and if $x \notin L$ then $f(x) = 1$. Since the class SPP is closed under complement, this will show that $L \in$ SPP. Define

$$f(x) = \Pi_{i=1}^{p(|x|)} \Pi_{\alpha \in F_q^*} (acc_{M'}(x, i, 0) - acc_{M'}(x, i, \alpha))$$

It is easy to see that $f \in$ GapP (See [FFK91] for the closure properties of the class GapP). To see that f has the above-mentioned behaviour, let $x \notin L$. Then $Sol_R(x) = \{\mathbf{0}\}$. In this case, for any pair $(i, \alpha), 1 \leq i \leq p(|x|)$ and $\alpha \in F_q^*$, $acc_{M'}(x, i, \alpha) = 0$. So the product evaluates to 1. Now, suppose $x \in L$. Then, from the definition of $\mathcal{L}_{VS}(q)$, it means that, there exist a $\mathbf{y} \in Sol_R(x)$, $i; 1 \leq i \leq p(|x|)$ and an $\alpha \in F_q^*$ such that $y_i = \alpha$. So, from Proposition 4.2, it follows that one of the terms in the product of the definition of f vanishes and hence $f(x) = 0$. This completes the proof. ■

5 NP Problems over Abelian Groups

In this section, we show that those NP problems whose solution set form a rght coset of an abelian black-box group is low for the class PP. As an interesting consequence of this result, we derive PP-lowness for various group theoretic problems over abelian groups. First, we formally define the notion of imposing the abelian group structure on the solution set of an NP problem. We define the relativised version of this notion.

Definition 5.1 Let $\mathcal{B}$ be a group family $\{B_m\}_{m \geq 1}$. For any $A \subseteq \Sigma^*$, let $\mathcal{L}_{rc}^{abel}(A, \mathcal{B})$ denote the class of languages $L \in NP(A)$ such that L is defined by a relation $R \in \mathrm{P}(A)$, and for every $x \in A$ the set $Sol_R(x)$ is a right coset of some *abelian* subgroup of $B_{i(|x|)}$, for a polynomial i. If $A = \phi$, we denote this class by $\mathcal{L}_{rc}^{abel}(\mathcal{B})$.

Before we go into the proof of the main theorem, we state results from the theory of finite abelian groups which are essential for the proofs in this section. For the proofs of standard group-theoretic results and basic definitions, we refer the reader to the text-books [Bu, Hall]. The following representation theorem is a fundamental result in the theory of finite abelian groups.

Theorem 5.2 [Bu] *Let G be a finite abelian group such that $|G| = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$, where the p_i's are distinct primes. The group G can be expressed as the direct product of its Sylow subgroups $S(p_1), S(p_2), \dots, S(p_r)$ where $|S(p_i)| = p_i^{e_i}$ for $1 \leq i \leq r$. Furthermore, for $1 \leq i \leq r$, each Sylow subgroup $S(p_i)$ can be uniquely expressed as the direct product of cyclic groups of orders $p_i^{e_{i1}}, p_i^{e_{i2}}, \dots, p_i^{e_{is_i}}$ such that $e_{i1} \geq e_{i2} \geq \dots \geq e_{is_i}$ and $\Sigma_{j=1}^{s_i} e_{ij} = e_i$. This decomposition of G is unique.*

Let p be a prime and consider an abelian p-group S of order p^e. The above theorem implies that there is a *unique* sequence of natural numbers $e_1 \geq e_2 \geq \dots \geq e_m$ such that $\Sigma_{1 \leq j \leq m} e_j = e$ and S can be expressed as a direct product of m cyclic groups of respective orders p^{e_j}, $1 \leq j \leq m$. The sequence $(e_1, e_2, \dots, e_m)$ is called the *type* of the p-group S. Let G be a finite abelian group. An element $g \in G$, $g \neq e$, is said to be *independent* of a set X of elements of G if $\langle g \rangle \cap \langle X \rangle = \langle e \rangle$. A generator set S of G is an *independent set of generators* for G if every $g \in S$ is independent of $S - \{g\}$. As a consequence of Theorem 5.2, all finite abelian groups have independent generator sets.

For the proof of the main theorem, we need to know the number of independent generator sets of an abelian group. In the next subsection, we derive a formula for counting the number of independent generator sets of an abelian group.

5.1 Number of Independent Generator sets

Let p be a prime. Let G be an abelian p-group. Let $G_\mu = \{g \in G \ : \ o(g) \leq p^\mu\}$. Then G_μ is a subgroup of G. The following lemma gives the cardinality of G_μ in terms of μ and the type $(m_1, m_2, \ldots, m_s)$ of G.

Lemma 5.3 [Bu] *Let G be a* p-group *of type $(m_1, m_2, \ldots, m_s)$. Let $G_\mu = \{g \in G \mid o(g) \leq p^\mu\}$ and $|G_\mu|$ be p^ν. Then $\nu = k\mu + \Sigma_{i=k+1}^{s} m_i$ where k is such that $m_k \geq \mu \geq m_{k+1}$.*

From the above lemma, we can deduce easily the following corollary.

Corollary 5.4 *Let G be a* p-group *of type $(m_1, m_2, \ldots, m_s)$.*

1. *The number of elements in G of order p^{m_1} is $\mid G_{m_1} \mid - \mid G_{m_1 - 1} \mid$.*

2. *Let $g_1, g_2, \ldots, g_s$ be an independent set of generators for G with $o(g_i) = p^{m_i}$, $1 \leq i \leq s$. Let G_i be the subgroup of G generated by the set $\{g_1, g_2, \ldots, g_i\}$. Then for every i, $1 \leq i \leq s$, the factor group $H^i = G/G_i$ is of type $(m_{i+1}, m_{i+2}, \ldots, m_s)$, and there are $|H^i_{m_{i+1}}| - |H^i_{m_{i+1}-1}|$ elements of order $p^{m_{i+1}}$ in H^i.*

Using the above two results we prove the following lemma which gives a formula for counting the number of independent generator sets of an abelian group.

Lemma 5.5 *Let G be a finite abelian p-group of type $(m_1, m_2, \ldots, m_s)$. Let $g_1, g_2, \ldots, g_s$ be an independent set of generators for G with $o(g_i) = p^{m_i}$, $1 \leq i \leq s$. Let G_i be the subgroup of G generated by the set $\{g_1, g_2, \ldots, g_i\}$. Let $H^i = G/G_i$. Then the number of independent generator sets K of G is given by $K = \prod_{i=0}^{s}(|H^i_{m_{i+1}}| - |H^i_{m_{i+1}-1}|)p^{im_{i+1}}$.*

Proof. Suppose we already have an independent set $\{g_1, g_2, \ldots, g_i\}$ of elements such that $o(g_j) = p^{m_j}$ for $1 \leq j \leq i$. We shall count the number of elements of order $p^{m_{i+1}}$ in G, which are independent to the set $\{g_1, g_2, \ldots, g_i\}$.

Firstly, from Corollary 5.4, it follows that there are $|H^i_{m_{i+1}}| - |H^i_{m_{i+1}-1}|$ elements (cosets of G_i in G) of order $p^{m_{i+1}}$ in H^i. So it follows that if we could count the number of elements of G of order $p^{m_{i+1}}$ in any right coset $G_i g$ of order $p^{m_{i+1}}$ in G, we can calculate the number of elements of G of order $p^{m_{i+1}}$ independent to the set $\{g_1, g_2, \ldots, g_i\}$.

Consider a right coset $G_i g$ of order $p^{m_{i+1}}$. It follows from the proof of Theorem 5.2 (described in [Bu]), that any right coset $G_i g$ of order $p^{m_{i+1}}$ has an element g_{i+1} of order $p^{m_{i+1}}$ in G independent of $\{g_1, g_2, \ldots, g_i\}$. So, $G_i g = G_i g_{i+1}$ and we want the number of elements $h \in G_i$ satisfying the eqation: $(hg_{i+1})^{p^{m_{i+1}}} = e$. Notice that, since g_{i+1} is independent of G_i, $(hg_{i+1})^j$ cannot be equal to identity for $j < p^{m_{i+1}}$. By Theorem 5.2 it follows that there exits indices $l_1, \ldots, l_i$; $l_j < p^{m_j}$ for $1 \leq j \leq i$ such that $h = g_1^{l_1} \ldots g_i^{l_i}$. Since $(g_{i+1})^{p^{m_{i+1}}} = e$ and the set $\{g_1, g_2, \ldots, g_i\}$ is independent, the number of elements $h \in G_i$ satisfying the condition $(hg_{i+1})^{p^{m_{i+1}}} = e$ is same as the number of tuples $(l_1, \ldots, l_i)$; $l_j < p^{m_j}$ for $1 \leq j \leq i$ such that $g_j^{l_j p^{m_{i+1}}} = e$ for $1 \leq j \leq i$. Since the order of g_j is p^{m_j}, the above condition holds for all those l_j such that p^{m_j} divides $l_j p^{m_{i+1}}$. So for each j; $1 \leq j \leq i$, there are $p^{m_{i+1}}$ possibilities. This implies that the total number of tuples $(l_1, \ldots, l_i)$; satisfying the condition $g_j^{l_j p^{m_{i+1}}} = e$ for $1 \leq j \leq i$ is $p^{im_{i+1}}$.

So the number of elements in G of order $p^{m_{i+1}}$ independent of the set $\{g_1, g_2, \ldots, g_i\}$ are $(|H^i_{m_{i+1}}| - |H^i_{m_{i+1}-1}|)p^{im_{i+1}}$. Hence the lemma follows from induction. ∎

Next we extend the above lemma to get the number of independent generators for an arbitrary abelian group.

Lemma 5.6 *Let G be a finite abelian group of order $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$. Let K_i be the number of independent generators of the p_i-Sylow subgroup of G. Then the number of independent generators K for G is given by $K = \prod_{i=1}^{k} K_i$*

5.2 Main Theorem

In this subsection, we prove that those languages in NP, whose solution set forms a right coset of some abelian black-box group, is low for the counting class PP. Before the theorem, we introduce some notations and state a weaker version of a theorem from [KST92] which we use for the proof of the main theorem.

Let M be an oracle NP machine and let $A \in$ NP accepted by an NP machine N. and let $f \in$ FP. We say that M^A make *1-guarded* queries if on any input x, M^A only asks queries y for which $N(y)$ has either 0 or 1 accepting path.

Theorem 5.7 ([KST92]) *Let M be an oracle NP machine. If $A \in$ NP such that for all inputs M^A makes only 1-guarded queries to A then the function $h(x) = gap_{M^A}(x)$ is in* GapP.

The next theorem gives us a sufficiant condition for a language to be low for PP. In order to prove lowness of any $L \in \mathcal{L}_{rc}^{abel}(\mathcal{B})$ for PP, we show the existance of a machine accepting L satisfying the conditions of the following theorem. The proof of this theorem is quite standard, and closely follows the line of the proof of the fact that the class LWPP is low for PP given in [FFK91]. We omit the proof.

Theorem 5.8 *Let L be a language such that there exists a* GapP *machine M' and a* GapP *function $f : \mathbf{N} \to \mathbf{Z}$ with the property: for all $x \in L$, $gap_{M'}(x) = f(1^{|x|})$ and for all $x \notin L$, $gap_{M'}(x) = 0$. Then L is low for* PP.

Theorem 5.9 *Let $\mathcal{B}$ be any group family. Then $\mathcal{L}_{rc}^{abel}(\mathcal{B})$ is low for* PP.

Proof. Let $\mathcal{B} = \{B_l\}_{l>0}$ be a group family. Let $L \in \mathcal{L}_{rc}^{abel}(\mathcal{B})$ witnessed by the NP machine M, such that for $x \in L$, $Sol_R(x)$ is a right coset of an abelian subgroup, say G_x, of $B_{r(|x|)}$, where r is some polynomial. Let $|B_{r(|x|)}| = n$ whose prime factorization is $p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$. Let $G^{(i)}$ is the p_i-Sylow subgroup of G_x. We first give an intuitive sketch of the proof and then describe the details.

Consider a nondeterministic machine N_A that starts its computation by simulating the NP machine that accepts L. If N_A could somehow compute the number $|Sol_R(x)| = |G_x|$ on each of its accepting paths, then it has to just branch into appropriate number of accepting paths, so that the total number of accepting paths only depends on $|x|$. So, by applying Theorem 5.8 we can show that L is low for PP.

Thus the crux of the proof is design a suitable method for computing $|G_x|$, and build this method into N_A. Observe that by Theorem 5.2 $|G_x|$ can be computed if the type of each Sylow subgroup of G_x is known. Therefore, N_A guesses the types of each of the Sylow subgroups of G_x. In order to authenticate the guessed types, N_A further guesses an independent set of generators for each of the Sylow subgroups of G_x and computes the orders of each generator. Here, Lemma 5.6 which gives us a formula for the number of independent generator sets of an abelian p-group of given type comes handy. Next, N_A is left with the task of verifying that each of the guessed generator sets are indeed independent. In order to do this verification N_A needs to use an oracle $\mathcal{I}$ (which is an NP language formally defined below). A delicate part of the design of N_A is that the queries it makes to $\mathcal{I}$ are 'guarded' in the following sense: for each query q to $\mathcal{I}$ made by N_A, it holds that the NP machine that accepts $\mathcal{I}$ has at most *one* accepting path. This 'guarded access' to $\mathcal{I}$ enables us to eventually do away with the oracle $\mathcal{I}$. This is accomplished by suitably applying Theorem 5.7.

Call a computation path of N_A *good* if along that path N_A guesses independent generator sets for each of the Sylow subgroups of G_x. One problem still left with N_A is that there may be computation paths along which one or more guessed independent generator set(s) generates only a *proper* subgroup of the corresponding Sylow subgroup of G_x. Call such paths as *bad* paths.

So, N_A actually accepts only the language defined as follows: $A = \{\langle x, w, p_i, m\rangle \mid w \in Sol_R(x)$; the p_i-Sylow subgroup $G^{(i)}$ of G_x has order divisible by $p_i^m\}$.

In order to get rid of the bad paths of N_A we design another machine N which simulates N_A in a suitable manner so that for each bad paths of N_A the machine N produces a zero *gap*, and N produces a fixed nonzero gap for each good path of N_A.

We shall now turn to the formal details of the proof. First we shall define a language in $\mathcal{I} \in NP$ which can be used to test independence of elments in an abelian group. Define $\mathcal{I} = \{\langle 0^m, S, g\rangle | S \subseteq B_m; g \in B_m; \langle S\rangle$ is abelian and $g \in S\}$

Claim 5.9.1 *There exists an NP-machine M whcih accepts $\mathcal{I}$. Moreover if S is an independent set of elements, then for those inputs $\langle 0^m, S, g\rangle \in \mathcal{I}$, M will have a unique accepting path.*

Proof. Firstly, we note that if the prime factorization $p_1^{e_1}p_2^{e_2}\dots p_k^{e_k}$ of $|B_m|$ is given, then $o(g)$ for $g \in B_m$, can be computed in time polynomial in m. To see this, observe that $o(g)$ can be expressed as $p_1^{d_1}p_2^{d_2}\dots p_r^{d_r}$, each $d_i \le e_i$. Clearly d_i is the smallest j such that $(g^{n/p_i^{e_i}})^{p_i^j} = e$. Since $1 \le d_i \le e_i$ and e_i is bounded by a polynomial in m, for every i, each d_i can be computed in time bounded by a polynomial in m.

The NP-machine M after verifying that $\langle S\rangle$ is abelian, computes $|B_m|$, guesses its prime factorization in a UP way [FeKo92], compute $o(h)$ for all $h \in S$, guesses indices $1 \le l_h \le o(h)$ and verifies that $g = \prod_{h\in S} h^{l_h}$. Now, if S is independent, then there will be a unique guess for l_h forall $h \in S$. So in this case M will have a unque accepting path. ∎

Next, we define a language $A = \{\langle x, w, p_i, m\rangle \mid w \in Sol_R(x)$; the p_i-Sylow subgroup $G^{(i)}$ of G_x has order divisible by $p_i^m\}$.

For an integer n, define positive integers $L_i(n) = p_i^{e_i^2+e_i}\prod_{1\le\alpha\le e_i}(p_i^\alpha - 1)$, for $1 \le i \le k$, where $|B_{r(n)}| = p_1^{e_1}p_2^{e_2}\dots p_k^{e_k}$.

Claim 5.9.2 *There is an oracle NP machine $N_A^{\mathcal{I}}$ that makes 1-guarded queries to $\mathcal{I}$ and accepts A. Furthermore, for inputs $\langle x, w, p_i, m\rangle$ in A, where $p_i^m = |G^{(i)}|$, $N_A^{\mathcal{I}}$ has exactly $L_i(|x|)$ accepting paths.*

Proof. We will define an oracle NP machine $N_A^{\mathcal{I}}$ that makes 1-guarded queries to $\mathcal{I}$ and accepts A with the desired properties.

The idea is that $N_A^{\mathcal{I}}$ on input $\langle x, w, p_i, m\rangle$, guesses a type $\langle m_{1,i}, m_{2,i}, \ldots, m_{k(i),i}\rangle$ and verifies that $m = \sum_{j=1}^{k(i)} m_{j,i}$. It then verifies that this is the type of a *subgroup* of $G^{(i)}$ by making suitable guarded queries to $\mathcal{I}$, (where $Sol_R(x) = G_x w$ is a right coset of an abelian subgroup G_x of $B_{r(|x|)}$).

DESCRIPTION OF MACHINE $N_A^{\mathcal{I}}(x, w, p_i, m)$

1 **if** $w \notin Sol_R(x)$
2   **then Reject**
3 **Guess** the UP certificate for the primality of p_i and check that p_i is prime
4 **if** p_i not pime
5   **then Reject**
6 Compute the following
7   (a) $n = |B_{r(|x|)}|$
8   (b) Index e_i which is the highest p_i power that divides n
9   (c) $L_i(r(|x|))$
10 (* *All the above quantities can be computed in time polynomial in* $|x|$ *)
11 **Guess** k: $1 \leq k \leq e_i$.
12 **Guess** the type $(m_1, m_2, \ldots, m_k)$ where $1 \leq m_1, m_2, \ldots, m_k \leq e_i$
13 **if** $(m \neq \sum_{j=1}^{k} m_j)$ **or** $(m_j > m_i$ for $j > i)$
14   **then Reject**
15 Using the type $(m_1, m_2, \ldots, m_k)$ Compute K_i as in Lemmas 5.6 and 5.3
16 **Guess** distinct strings $g_1, \ldots, g_k \in \Sigma^{r(|x|)}$
17 **if** $(o(g_j) \neq p_i^{m_j})$ **or** $(g_j w \notin Sol_R(x))$
18   **then Reject**
19 $j \leftarrow 2$
20 **while** $\langle o^{r(|x|)}, \{g_1, \ldots, g_{j-1}\}, g_j\rangle \notin \mathcal{I}$ **and** $j \leq k$
21   **do** $j \leftarrow j + 1$
22 **if** $j = k + 1$
23   **then** Branch into $L_i(|x|)/K_i$ paths and **Accept**
24   **else Reject**

It is easy to see that $N_A^{\mathcal{I}}$ accepts A. To see that the rest of the claim is correct, we first note that if the **while**-loop in *lines* 20-21 is exited with $j = k + 1$, then the **while**-loop condition guarantees that the guessed elements $g_1, g_2, \ldots, g_k$ are independent. *lines* 17-18 guarantees that each g_j has order p^{m_j} and are elements of $G^{(i)}$, $1 \leq j \leq k$. Also *lines* 13-14 ensures that $m = \sum_{j=1}^{k} m_j$. Therefore $\langle g_1, g_2, \ldots, g_k\rangle$ is a subgroup of $G^{(i)}$ of order p_i^m.

Let us consider the case when $p_i^m = |G^{(i)}|$. If the **while**-loop in *lines* 20-21 is exited with $j = k+1$, then it holds that $\{g_1, g_2, \ldots, g_k\}$ is a generator set for $G^{(i)}$ and, moreover, *lines* 22-23 give rise to $L_i(|x|)/K_i$ accepting paths. Furthermore, Theorem 5.2 guarantees that there is only a unique guess of the type $m_1 \geq m_2, \ldots \geq m_k$ that can lead to accepting paths. For that unique guess K_i is correctly computed in *line* 15. Also, from Lemma 5.6, we know that $G^{(i)}$ has K_i generator sets. Thus $N'^{\mathcal{I}}$ has $K_i \cdot L_i(|x|)/K_i = L_i(|x|)$ accepting paths on input $\langle x, w, p_i, m\rangle$ when p_i^m is $|G^{(i)}|$.

To see that the machine $N_A^{\mathcal{I}}$ is making 1-guarded queries to $\mathcal{I}$, notice that $N_A^{\mathcal{I}}$ is querying $\mathcal{I}$ only in *line*-20. Suppose at the j^{th} iteration of the **while**-loop, for a query $\langle o^{r(|x|)}, \{g_1, \ldots, g_{j-1}\}, g_j\rangle$ to $\mathcal{I}$, the machine M described in 5.9.1 has more than one accepting path. This means that the set $\{g_1, \ldots, g_{j-1}\}$ is not independent. That is there exists g_i; $i < j$ such that $g_i \in \langle g_1, \ldots, g_{i-1}\rangle$. This is a contradiction to the fact that the machine has not exited the **while**-loop at the i^{th} iteration. Hence the queries made by $N_A^{\mathcal{I}}$ to $\mathcal{I}$ are 1-guarded. ■

Now, using the Theorem 5.7, we can replace the oracle machine $N_A^{\mathcal{I}}$ with a GapP machine N'_A, which produces the same gap as the number of accepting paths of $N_A^{\mathcal{I}}$. The following claim is a direct application of Theorem 5.7 to Claim 5.9.2.

Claim 5.9.3 *There is a* GapP *machine* N'_A *defining a* GapP *function* h *such that the language* A *is* $\{\langle x, w, p_i, m\rangle \mid h(\langle x, w, p_i, m\rangle) \neq 0\}$. *Furthermore, for inputs* $\langle x, w, p_i, m\rangle$ *in* A, *if* $p_i^m = |G^{(i)}|$ *then* $h(\langle x, w, p_i, m\rangle) = L_i(|x|)$.

The idea of the remainder of the proof is as follows. We define a GapP machine N which on input $x \in \Sigma^*$ has the following accepting behaviour. For all $x \in L$, N will have a *gap* which depends only on $|x|$ and for all $x \notin L$, N will produce a 0 *gap*. Once existance of such a machine for L is being proved, it follows that $L \in$ LWPP.

Claim 5.9.4 *There exists a* GapP *machine* N *which on input* $x \in \Sigma^*$ *has the following accepting behaviour. For all* $x \in L$, N *will produce a gap* $= n\prod_{i=1}^{k} L_i^{e_i}(|x|)$ *and for all* $x \notin L$, N *will produce a gap* $= 0$.

We shall first give the description of machine N. Then we shall prove that N has the above mentioned accepting behaviour.

DESCRIPTION OF $N(x)$

Compute $\mid B_{r(|x|)} \mid = n$
Guess $k : 1 \leq k \leq \log(n)$
for $i \leftarrow 1$ **to** k
    do **Guess** integer $p_i > p_{i-1}$ and UP certificate s_i of primality of p_i
        if p_i not prime
          then produce a *gap*=0
        Guess index e_i
if $n \neq p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$
  then produce a *gap*=0
Guess $w \in \Sigma^{r(|x|)}$
if $w \notin Sol_R(x)$
  then produce a *gap*=0
for $1 \leftarrow 1$ **to** k
    do Guess $l_i : 1 \leq l_i \leq e_i$
Compute $\beta = \prod_{i=1}^{k} p_i^{l_i}$
Branch into n/β paths and produce a gap of
$\prod_{i=1}^{k}[h(\langle x, w, p_i, l_i\rangle) \cdot L_i^{l_i - 1}(x) \cdot \prod_{m=l_i+1}^{e_i}(L_i(x) - h(\langle x, w, p_i, m\rangle))]$

From the definition of the group family, it follows that the computation at *line* 1 is polynomial time. Since, checking whether an integer is prime or not is in the class UP [FeKo92], *line* 2-10 of the machine ensures that after the computation of *line* 10, N will have the unique prime factorization of n on exactly one of its path and all other paths will produce a *gap*=0. Now, we shall see the accepting behaviour of M on this path.

Recall that h is the GapP function computed by N'_A defined in Claim 5.9.3. Consider the gap $\prod_{i=1}^{k}[h(\langle x, w, p_i, l_i\rangle) \cdot L_i^{l_i-1}(x) \cdot \prod_{m=l_i+1}^{e_i}(L_i(x) - h(\langle x, w, p_i, m\rangle))]$ that machine N produces in *lines* 9-10. Let us denote the gap by X, for brevity. Consider the case when, for each $1 \leq i \leq k$, the value l_i guessed in *lines* 6-7 is m_i, where $p_i^{m_i}$ is the size of the p_i-Sylow subgroup $G^{(i)}$ of G_x. Then it holds that $p_i^{l_i} = |G^{(i)}|$ for each i, $1 \leq i \leq k$. In this case $h(\langle x, w, p_i, l_i\rangle) = h(\langle x, w, p_i, m_i\rangle) = L_i(x)$ for all $1 \leq i \leq k$ (Claim 5.9.3) and $h(\langle x, w, p_i, m\rangle) = 0$ for all $m > l_i$. So the gap corresponding to this particular set of guesses for l_i, $1 \leq i \leq k$, is $\prod L_i(x)^{e_i}$. On the other hand, consider the case when one of the guessed values l_i is different from the corresponding correct value m_i. For that index i, if $l_i > m_i$ then $h(\langle x, w, p_i, l_i\rangle) = 0$, and if $l_i < m_i$ then $\prod_{m=l_i+1}^{e_i}(L_i(x) - h(\langle x, w, p_i, m\rangle)) = 0$ since $L_i(x) - h(\langle x, w, p_i, m_i\rangle) = 0$, and $m_i \geq l_i + 1$. Thus, for an incorrect guess $l_1, l_2, \ldots, l_k$, the corresponding gap X will take the value 0. Therefore, the gap X evaluates to 0 (for each of the n/β paths into which it branches at *line* 9) if the guess $l_1, l_2, \ldots, l_k$ is incorrect. On the other hand, the gap X evaluates to $\prod L_i^{e_i}$ for the unique correct guess $m_1, m_2, \ldots, m_k$. Thus, the total gap produced under the correct guess of $m_1, m_2, \ldots, m_k$ is $|G_x| \cdot (n/|G_x|) \cdot \prod L_i^{e_i}$ which is $n \prod L_i^{e_i}$. The claim follows. ∎

Now to complete the proof of the theorem, we note that it is easy to design a *gap* machine which on input $1^{|x|}$ produces a gap $n \prod L_i^{e_i}$, where $n = |B_{r(|x|)}|$. So by Theorem 5.8 and Claim 5.9.4, it follows that L is low for PP. ∎

It is routine to verify that the above proof relativizes. We state a relativized version of the theorem.

Theorem 5.10 *Let $\mathcal{B}$ be any group family. Then for any language A, $\mathcal{L}_{rc}^{abel}(A, \mathcal{B})$ is low for* PP(A).

5.3 Application of the Main Theorem

Now we shall see the consequence of the above theorem on the PP-lowness for several abelian black-box groups problems. These problems are well-studied in computational group theory [CoFi92a, BaSz85, KST92, Ba92]. Here we would like to remark that in another paper [AV96], we actually show that all the problems we are considering here are low for PP, using proof methods specific to these problems. What is interesting here is that Theorem 5.9 places various abelian group problems in a uniform framework, and their PP-lowness follows directly.

The problems that we are interested in are membership testing, group factorisation, coset inetersection and double coset factorization. We formally define these problems for general black-box groups, over an arbitrary group family $\mathcal{B}$. We are interested in the case when these groups are abelian.

Rest-Memb $\stackrel{\text{def}}{=} \{\langle 0^m, S, g\rangle | S \subseteq B_m; g \in B_m; |S| \leq c(m)$ for a polynomial c; $g \in \langle S\rangle\}$

Memb $\stackrel{\text{def}}{=} \{\langle 0^m, S, g\rangle | S \subseteq B_m; g \in B_m; g \in \langle S\rangle\}$

Group-Fact $\stackrel{\text{def}}{=} \{(0^m, S_1, S_2, g) \mid S_1, S_2 \subseteq B_m \text{ and } g \in \langle S_1\rangle\langle S_2\rangle\}$.
Coset-Inter $\stackrel{\text{def}}{=} \{(0^m, S_1, S_2, g) \mid S_1, S_2 \subseteq B_m \text{ and } \langle S_1\rangle g \cap \langle S_2\rangle \neq \emptyset\}$.
Double-Coset-Memb $\stackrel{\text{def}}{=} \{(0^m, S_1, S_2, g, x) \mid S_1, S_2 \subseteq B_m \text{ and } g \in \langle S_1\rangle x\langle S_2\rangle\}$.

We first show that restricted membership testing for abelian groups (Rest-memb) over group family $\mathcal{B}$ is in $\mathcal{L}_{rc}^{abel}(\mathcal{X})$ for a suitably chosen group family $\mathcal{X}$ hence low for PP. Then we show that member ship testing (Memb)is polynomial time Turing reducible to Rest-Memb there by implying that Memb is low for PP.

Proposition 5.11 *Let $\mathcal{B}$ be a group family. Then restricted membership testing* (Rest-Memb) *for abelian groups is in $\mathcal{L}_{rc}^{abel}(\mathcal{B})$.*

Proof Sketch. By definition of group families, we know that $|B_m| = r$ is computable in time polynomial in m. Let $\mathbf{Z}_r^+$ denote the additive group of integers modulo r. Consider the set $\{\langle e_1, e_2, \ldots, e_{c(m)}\rangle \mid g_1^{e_1} g_2^{e_2} \ldots g_k^{e_k} = x \text{ and } e_i \in \mathbf{Z}_r^+ \text{ for } 1 \leq i \leq c(m)\}$. Clearly, we can define a polynomial-time computable relation such that this set is the solution set for the instance $(x, \langle g_1, g_2, \ldots g_k\rangle)$. It is easy to see that this set is a right coset of the group $G = \{\langle e_1, e_2, \ldots, e_{c(m)}\rangle \mid g_1^{e_1} g_2^{e_2} \ldots g_k^{e_k} = id\}$ which is a subgroup of the product group $\mathbf{Z}_r^+ \times \mathbf{Z}_r^+ \ldots \times \mathbf{Z}_r^+$ (the product taken $c(m)$ times). Let the group family consisting of these product groups be denoted $\mathcal{X}$. Then, it can be easily seen that the restricted abelian group membership testing problem is in $\mathcal{L}_{rc}^{abel}(\mathcal{X})$. ∎

It immediately follows from the above theorem and Proposition 5.11 that the restricted abelian group membership problem is low for PP. We state this as next corollary.

Corollary 5.12 *Let $\mathcal{B}$ be agroup family. Then restricted membership testing (*Rest-Memb) for abelian groups is low for PP.

Proposition 5.13 *Let $\mathcal{B}$ be a group family. Then the language* Memb *is polynomial-time Turing reducible to* Rest-Memb.

Proof. Let $(0^m, S, g)$ be an instance of the membership testing problem. From the definition of group families there is a polynomial p such that $|B_m| \leq 2^{p(|x|)}$. Using the following polynomial-time oracle machine with the Rest-Memb($\mathcal{B}$) as oracle, we can construct a generator set for A whose size is at most $p(|x|)$, and thereby reduce membership testing to restricted membership testing.

```
REDUCE((0^m, {g_1, ..., g_k}, g))
X ← {g_1}
for i ← 2 to k
    do if (0^m, X, g_i) ∉ Rest-Memb(B)
         then X ← X ∪ {g_i}
if (0^m, X, g) ∈ Rest-Memb(B)
  then  Accept
```

Clearly the set X constructed by the for-loop in the above algorithm generates $\langle\{g_1, \ldots, g_k\}\rangle$. Furthermore, in the above algorithm, each time we include an element in X, by Lagrange's theorem, the size of the corresponding subgroup generated by it at least doubles in cardinality. Since $|B_m| \leq 2^{p(|x|)}$, it must hold that $|X| \leq p(|x|)$. This completes the proof. ■

The next corollary immediately follows.

Corollary 5.14 *Let $\mathcal{B}$ be a group family. Then* Memb *for abelian groups is low for PP.*

Theorem 5.15 *Let $\mathcal{B}$ be a group family. Then the problems* Group-Fact, Coset-Inter *and* Double-Coset-Inter *with respect to $\mathcal{B}$, for abelian groups are low for PP.*

Proof. We prove the result only for Group-Fact. The results for Coset-Inter and Double-Coset-Inter can be proved similarly.

Consider an instance $(0^m, S_1, S_2, g)$ of Group-Fact for abelian groups with respect to the group family $\mathcal{B}$, where $\langle S_1 \rangle$ and $\langle S_2 \rangle$ are abelian subgroups of B_m.

Consider the relation $R = \{(g_1, S_1, S_2, g) \mid g_1 \in \langle S_1 \rangle \text{ and } g_1^{-1}g \in \langle S_2 \rangle\}$. It is easy to see that $R \in \mathrm{P(Memb)}$, for, the polynomial time machine has to query Memb for $(0^n, S_1, g_1)$ and $(0^n, S_2, g_1^{-1}g)$. Further, observe that corresponding to the relation R, the solution set of an instance $x = (0^m, S_1, S_2, g)$ is $Sol_R(x) = \{g_1 \mid g_1 \in \langle S_1 \rangle \text{ and } g_1^{-1}g \in \langle S_2 \rangle\}$. It is easy to see that $Sol_R(x)$ is a right coset of $\langle S_1 \rangle \cap \langle S_2 \rangle$ which is also an abelian group since $\langle S_1 \rangle$ and $\langle S_2 \rangle$ are abelian. This implies that Group-Fact is in $\mathcal{L}_{rc}^{abel}(\mathrm{Memb}, \mathcal{B})$. So from Theorem 5.10, it follows that Group Factorization is in PP(Memb). But from Corollary 5.14 we know that PP(Memb) = PP. Hence the theorem. ■

References

[AV96] V. Arvind and N. V. Vinodchandran. Solvable Black-Box Group Problems are low for PP *Symposium on Theoretical Aspects of Computer Science*, LNCS Vol: 1046, 99-110, 1996.

[Ba92] L. Babai. Bounded round interactive proofs in finite groups. *SIAM Journal of Discrete Mathematics*, 5: 88-111, (1992)

[Ba85] L. Babai. Trading group theory for randomness. *17th ACM Symposium on Theory of Computing*, 421-429, 1985.

[BaMo88] L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36: (1988), 254-276.

[BLS88] L. Babai, E. Luks and Á. Seress. Fast management of permutation groups. *Proc. 28th IEEE Symposium on Foundations of Computer Science,* 1988, 272-282.

[BaSz85] L. Babai and M. Szemerédi. On the complexity of matrix group problems I. *Proc. 25th IEEE FOCS*, 1984, 229-240.

[BHZ87] R. Boppana, J. Hastad and S. Zachos. Does coNP have short interactive proofs? Information Processing Letters, 25 (1987), 127-132.

[Bu] W. Burnside. *Theory of Groups of Finite Order*, Dover Publications, INC, 1955.

[CoFi92] G. Cooperman and L. Finkelstein. Random algorithms for permutation groups. CWI Quarterly, 5 (2) 1992, 93-105.

[CoFi92a] G. Cooperman and L. Finkelstein. Combinatorial tools for computational group theory. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 11, 1993.

[FFK91] S. Fenner, L. Fortnow, S. Kurtz. Gap-definable counting classes. *Proc. 6th Structure in Complexity Theory Conference*, 1991, 30-42.

[FeKo92] M. Fellows and N. Koblitz. Self-witnessing polynomial time complexity and prime factorization. *Proc. 6th Structure in Complexity Theory Conference*, 1992, 107-110.

[FHL79] M. Furst, J. E. Hopcroft and E. Luks. Polynomial time algorithms for permutation groups. *Proc. 21st IEEE Symposium of Foundations of Computer Science*, 1980, 36-45.

[GMR89] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proofs. SIAM Journal of Computing, 18(1) (1989), 186-208.

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

[Ho82] C. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism.* Lecture Notexs in Computer Science #136, Springer, 1982

[HK] K. Hoffmann and R. Kunz. *Linear Algebra.* Prentice Hall Inc., 1971.

[KST92] J. Köbler, U. Schöning, J. Torán. Graph isomorphism is low for PP. Journal of Computational Complexity, 2 (1992), 301-310.

[Kan92] W. M. Kantor. Notes on polynomial-time group theory. CWI Quarterly, 5 (2) 1992, 93-105.

[Ma79] R. Mathon. A note on graph isomorphism counting problem. Information Processing Letters 8 (1979), 131-132.

[Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. Journal of Computer and System Sciences, 37 (1988), 312-323.

[To91] PP is as hard as polynomial-time hierarchy. SIAM Journal on Computing, 20(5) (1991), 865-877.