Introduction

In this thesis we explore the algorithmic complexity of some algebraic problems in both the commutative and the noncommutative settings. Our motivation is to better understand algorithmic questions in both the settings and to see the interplay between them.

Commutative and noncommutative computation exhibit an interesting difference in computational complexity. For example, in the commutative setting computing the determinant has very efficient parallel algorithms (e.g. [MV97]). These algorithms actually describe polynomial-size algebraic branching programs for computing the determinant polynomial. On the other hand, in the noncommutative setting, Nisan [N91] has shown exponential size lower bounds on the size of any algebraic branching program which computes the determinant. In fact, more recently [AS10] it is shown that it is unlikely that the noncommutative determinant has a polynomial-sized arithmetic circuit; indeed, the existence of such a circuit would imply that the noncommutative permanent polynomial has a polynomial-sized arithmetic circuit, which in turn gives a polynomial-sized arithmetic circuit for the commutative permanent polynomial, which is widely believed to be false.

To see another example of the contrast between these two models consider the polynomial identity testing problem for algebraic branching programs and arithmetic circuits. For algebraic branching programs, this problem has a deterministic polynomial time algorithm [RS05] in the noncommutative setting, whereas in the commutative setting,
getting such an algorithm is a long standing open problem. In the case of arithmetic circuits, there is a randomized polynomial time algorithm based on Schwartz-Zippel lemma for the problem in the commutative setting whereas in the noncommutative setting, the problem can be solved in randomized polynomial time only if the polynomial computed by the given arithmetic circuit has polynomial (in the number of input variables) degree.

In this thesis we pursue this direction of research further and compare the complexities of various algebraic problems in the commutative and the noncommutative domains. We also investigate the possibility of applying the techniques and tools developed in the one model to the other. Specifically, we focus on the computational complexity of the problems over integer lattices, permutation groups and arithmetic circuits. Now we describe the main results in this thesis.

1.1 Sieving Algorithms for Lattice Problems

Lattices are geometric objects that can be pictorially described as the set of intersection points of an infinite regular grid in \( n \) dimensions. More precisely, given linearly independent vectors \( b_1, \ldots, b_n \in \mathbb{R}^n \) the lattice \( \mathcal{L} \) generated by them is the set of all integer linear combinations of \( b_i \)’s i.e. \( \mathcal{L} = \{ \sum_{i=1}^{n} \alpha_i b_i | \alpha_i \in \mathbb{Z} \} \). Despite their apparent simplicity, lattices have a rich combinatorial structure which leads to numerous applications in mathematics and computer science.

Two fundamental algorithmic problems concerning integer lattices are the shortest vector problem (SVP) and the closest vector problem (CVP). Given a lattice \( \mathcal{L} \subset \mathbb{R}^n \) by a basis, the shortest vector problem (SVP) is to find a shortest non-zero vector in \( \mathcal{L} \) with respect to a given metric. Likewise, the closest vector problem (CVP) takes as input a lattice \( \mathcal{L} \subset \mathbb{R}^n \) and a vector \( v \in \mathbb{R}^n \) and asks for a \( u \in \mathcal{L} \) closest to \( v \) with respect to a given metric.

The study of lattices from the computational point of view was marked by a major breakthrough: The LLL algorithm developed by Lenstra, Lenstra and Lovasz [LLL82] which gives an approximate solution for SVP in \( n \) dimensions. Given a rank \( n \) integer lattice, in deterministic polynomial time the LLL algorithm outputs a nonzero vector
$v \in \mathcal{L}$ whose $\ell_2$ norm is guaranteed to be within a $2^{O(n)}$ factor of the norm of a shortest nonzero vector in $\mathcal{L}$. The LLL algorithm also can be used to solve CVP within a $2^{O(n)}$ factor [Bab86].

Despite the relatively poor quality of the approximate solution in the worst case, the LLL algorithm allows us to devise polynomial time algorithms for various problems including polynomial factorization over rationals, breaking a knapsack-based cryptosystem, solving integer linear programs in a fixed number of variables etc [LLL82, MG02]. There are some results which improve the approximation factor in the LLL solution to a slightly sub-exponential factors (e.g. [Sch94]).

NP-hardness of CVP (in any $\ell_p$ norm) and SVP (in $\ell_\infty$ norm) was originally proved by van Emde Boas in 80’s [Bos81]. In fact, it is known that, even finding an approximate solution for CVP is a hard problem (e.g. [ABSS97], [DKS98]). In [DKS98] it is shown that CVP is NP-hard to approximate within approximation factor of $2^{O(\log n \log \log n)}$. Showing NP-hardness for CVP for polynomial approximation factor is an important open problem. The NP-hardness of SVP was conjectured in [Bos81] and remained probably the biggest open problem in the area for almost two decades. In a breakthrough paper Ajtai [Ajt98] proved that SVP is NP-hard under randomized reduction. In the recent years the hardness of approximating SVP is being explored, we know that SVP is hard to approximate within almost polynomial factor based on reasonable complexity theoretic assumption (see e.g. [HR07]).

Another line of research is to find efficient exact algorithm to solve CVP and SVP. The LLL algorithm enables us to solve SVP for constant-dimensional lattices in polynomial time. The fastest known deterministic algorithms to solve SVP or CVP exactly with respect to $\ell_p$ norm have running time $2^{O(n \log n)}$ ([Kan87], [Bl00]). In a seminal paper [AKS01] Ajtai, Kumar and Sivakumar gave a $2^{O(n)}$ time randomized exact algorithm for SVP for $\ell_2$ norm. Subsequently, in [AKS02] they gave a $2^{O(n)}$ time randomized algorithm to find a $1 + \epsilon$ approximate solution for CVP, for any constant $\epsilon > 0$. Their algorithms are based on a generic sieving procedure.

Another problem recently studied is the subspace avoiding problem. Given a $k$-dimensional subspace $M \subseteq \mathbb{R}^n$ and a full rank integer lattice $\mathcal{L} \subseteq \mathbb{Q}^n$, the subspace avoiding problem SAP [BN07], is to find a shortest vector in $\mathcal{L} \setminus M$. If subspace $M$ is zero dimensional
then clearly such SAP instance exactly captures SVP. Blömer and Naewe showed that CVP also reduces to SAP. On the other hand, Micciancio [Mi08] showed that CVP is equivalent to several other lattice problems including shortest independent vector problem (SIVP), successive minima problem (SMP) and subspace avoiding problem (SAP) under deterministic polynomial time rank-preserving reductions. In particular, the reductions in [Mi08] imply a $2^{O(n \log n)}$ time exact algorithm for SAP, SMP and SIVP.

In a breakthrough result Micciancio and Voulgaris [MV10] gave a $2^{O(n)}$ deterministic algorithm to solve many important lattice problems including CVP, SAP, SMP and SVP with respect to $\ell_2$ norm. Their algorithm crucially uses the fact that Voronoi cell of a lattice is convex when concerned metric is $\ell_2$ norm. For the general $\ell_p$ norms the Voronoi cell need not be convex. Very recently, via a clever ellipsoid covering technique Dadush, Piekert and Vempala [DPV10] have extended this result to all $\ell_p$ norms.

**Results in this thesis**

In the work presented in this thesis we focus on the application of AKS sieving to the problem of finding an exact solution for SAP and CVP with respect to general $\ell_p$ norms.

We know that SAP is a generalization of both SVP and CVP. When dimension of the input subspace is zero, it exactly captures SVP for which we have exact $2^{O(n)}$ algorithm [AKS01]. So it is natural to explore the complexity of SAP as the dimension of the subspace increases. Given a rank $n$ integer lattice $\mathcal{L}$ and a subspace $M \subset \mathbb{R}^n$ of dimension $k$ we give a $2^{O(n+k \log k)}$ algorithm to solve SAP with respect to $\ell_p$ norm. This algorithm is based on the AKS sieving. [BN07] also give an algorithm for SAP based on the AKS sieving. Our algorithm performs better, parameterized on the dimension of the subspace because in our analysis we exploit the coset structure of the lattice $\mathcal{L} \cap M$ inside $\mathcal{L}$. This enable us to sample lattice points from a coset of a shortest vector in $\mathcal{L} \setminus M$ and apply packing argument within the coset. As applications of this algorithm we obtain the following results:

- We show that given a full rank lattice $\mathcal{L} \subset \mathbb{Q}^n$ there is $2^{O(n)}$ time randomized algorithm to compute linearly independent vectors $v_1, v_2, \ldots, v_i \in \mathcal{L}$ such that $\|v_i\|_p = \lambda_i^p(\mathcal{L})$ if $i$ is $O\left(\frac{n}{\log n}\right)$, where $\lambda_i^p(\mathcal{L})$ denotes the $i^{th}$ successive minima of
Chapter 1. Introduction

\( \mathcal{L} \) with respect to \( \ell_p \) norm. Given a full rank lattice \( \mathcal{L} \subset \mathbb{Q}^n \) and \( v \in \mathbb{Q}^n \) we also give a \( 2^{O(n)} \) time algorithm to solve CVP(\( \mathcal{L}, v \)) if the input \((v, \mathcal{L})\) fulfills the promise \( d(v, \mathcal{L}) \leq \frac{\sqrt{3}}{2} \lambda_{O(\log n)}(\mathcal{L}) \).

- We show that CVP with respect to \( \ell_p \) norm can be solved in \( 2^{O(n)} \) time if there is a \( 2^{O(n)} \) time algorithm to compute a closest vector to \( v \) in \( \mathcal{L} \) where \( v \in \mathbb{Q}^n \), \( \mathcal{L} \subset \mathbb{Q}^n \) is a full rank lattice and \( v_1, v_2, \ldots, v_n \in \mathcal{L} \) such that \( \|v_i\|_p \) is equal to \( i^{th} \) successive minima of \( \mathcal{L} \) for \( i = 1 \) to \( n \) are given as an additional input to the algorithm. As a consequence, we can assume that successive minima are given for free as an input to the algorithm for CVP.

- We give a new \( 2^{O(n+k \log 1/\epsilon)} \) time randomized algorithm to get a \( 1+\epsilon \) approximate solution for SAP, where \( n \) is the rank of the lattice and \( k \) is the dimension of the subspace. We get better approximation guarantee than the one in [BN07] parameterized on \( k \).

- We show that the AKS-sieving not only works for all \( \ell_p \) norms, but also for a more general notion of norm specified by a gauge function [Si45] and we need only oracle access to the gauge function.

The results presented in this chapter appeared in [AJ08b].

1.2 Algorithmic Problems for metrics on Permutation Groups

Motivated by the generic nature of the AKS-sieving procedure, it is natural to ask whether it can work for similar optimization problems in the other domains. Specifically, we investigate the computational complexity of the two natural problems for metrics on permutation groups given by generating sets (a noncommutative domain).

Given a metric \( d \) on the symmetric group \( S_n \), the weight of a permutation \( \pi \in S_n \) with respect to \( d \) is \( w_d(\pi) = d(\pi, e) \) where \( e \) denotes the identity permutation. Given a permutation group \( G = \langle A \rangle \leq S_n \) by a generating set \( A \) of permutations, we explore the algorithmic complexity of the minimum weight problem (denoted MWP) and the
subgroup distance problem (denoted SDP) for natural permutation metrics. Given a
permutation group $G \leq S_n$ by a generating set $A$ and $k \in \mathbb{R}^+$, then for a metric $d$
on the permuta\nation $\pi \in G$, and $k \in \mathbb{R}^+$ the subgroup distance problem with respect to a metric $d$ is to check if there exist a permutation $\rho \in G$ such that $d(\pi, \rho) \leq k$. These problems were studied by Cameron et al. in [BCW06, CW06] and shown to be NP-complete for several natural permutation metrics.

These problems are analogous to the shortest vector problem and the closest vector problem for integer lattices, and to the minimum Hamming weight problem and nearest codeword problem for linear codes. The corresponding problems for lattices and codes are NP-hard, and their approximability is a subject of current intensive study (see e.g. [MG02]). Our primary motivation stems from the fact that lattices and codes are abelian groups, and it is interesting to ask if the upper and lower bound techniques and results for approximability can be extended to arbitrary (nonabelian) permutation groups.

Results in this thesis

We study the complexity of MWP with respect to Hamming and $\ell_\infty$ metrics. Hamming distance between permutations $\tau, \pi \in S_n$ is defined as $d(\tau, \pi) = |\{i | \tau(i) \neq \pi(i)\}|$. $\ell_\infty$ distance between $\tau, \pi$ is $d(\tau, \pi) = max_{1 \leq i \leq n}|\tau(i) - \pi(i)|$. MWP is NP-hard with respect to both of these metrics even for abelian permutation groups.

A naive brute-force search algorithm for MWP (which enumerates all the permutations and finds a permutation in $G$ with shortest nonzero norm) can take up to $n!$ steps since $G \leq S_n$ can have up to $n!$ elements. It easily follows that if $G \leq S_n$ is an abelian group then $|G| \leq 2^{O(n)}$, so using classical Schrier-Sims algorithm for finding pointwise stabilizer subgroups of permutation groups ([Lu93]) we can enumerate all the permutations in $G$ and find one with the smallest nonzero norm. This gives a $2^{O(n)}$ algorithm to solve MWP for abelian groups.

More interesting case is that of the nonabelian permutation groups. In the case of Hamming metric we give a deterministic $2^{O(n)}$ time algorithm which is group theoretic in
nature. The algorithm is based on the classical Schrier-Sims algorithm. However, the problem for \( l_\infty \) metric does not appear amenable to a permutation group-theoretic approach. We give a \( 2^{O(n)} \) time randomized algorithm for the problem. Interestingly, for this algorithm we are able to adapt ideas from the Ajtai-Kumar-Sivakumar algorithm for the shortest vector problem for integer lattices [AKS01]. Other results presented in the thesis include:

- It is known that SDP is NP-hard ([BCW06]) and it easily follows that SDP is hard to approximate within a factor of \( \log^{O(1)} n \) unless \( P=NP \). In contrast, we show that SDP for approximation factor more than \( n/\log n \) is not NP-hard unless there is an unlikely containment of complexity classes.

- For several permutation metrics, we show that the minimum weight problem is polynomial-time reducible to the subgroup distance problem for solvable permutation groups.

These results adapts ideas from the analogous results in the case of integer lattices. The results presented in this chapter appeared in [AJ08a].

1.3 Arithmetic Circuits, Branching Programs and Monomial Algebras

In this part of the thesis we explore the polynomial identity testing problem and certain lower bound questions for arithmetic circuits and algebraic branching programs. Superpolynomial lower bounds for the size of commutative arithmetic circuits or algebraic branching programs for explicit polynomials is one of the most challenging open problem in arithmetic circuit complexity. Lower bounds are known only for some of the special classes of the commutative arithmetic circuits like depth 3 circuits, some restricted classes of depth 4 circuits etc. The general lower bound question is still unsolved despite the efforts of several researchers.

In the noncommutative case the question is better understood. Nisan in the pioneering paper [N91] studied the lower bounds for the noncommutative computation. Using
a rank argument Nisan showed that the Permanent and the Determinant polynomials in the *free* noncommutative ring $\mathbb{F}\{x_{11}, \ldots, x_{nn}\}$ require exponential size noncommutative formulas (and the noncommutative algebraic branching programs). Chien and Sinclair [CS04] explored the same question over other noncommutative algebras. They refined Nisan’s rank argument to show an exponential size lower bounds for formulas computing the Permanent or the Determinant over the algebra of $2 \times 2$ matrices over $\mathbb{F}$, the quaternion algebra, and several other interesting examples.

In a similar spirit as [CS04], in the work presented in this thesis we explore the lower bound question over other noncommutative algebras. An ideal $I$ of the noncommutative polynomial ring $\mathbb{F}\{x_1, \ldots, x_n\}$ (which we denote by $\mathbb{F}\{X\}$ when the set of indeterminates is clear from the context) is a subring that is closed under both left and right multiplication by the ring elements. The circuit complexity of the polynomial $f$ in the quotient algebra $\mathbb{F}\{X\}/I$ is $C_I(f) = \min_{g \in I} C(f + g)$ where for $h \in \mathbb{F}\{X\}$, $C(h)$ is the circuit complexity of $h$ over free noncommutative algebra $\mathbb{F}\{X\}$. We can define the algebraic branching program complexity of a polynomial over $\mathbb{F}\{X\}/I$ analogously. We study the question of proving lower bound on the arithmetic circuit complexity and the algebraic branching program complexity of explicit polynomials over quotient algebra $\mathbb{F}\{X\}/I$ where the ideal $I$ is given by generating set of polynomials.

If the ideal $I$ is generated by monomials in $\mathbb{F}\{X\}$ the algebra $\mathbb{F}\{X\}/I$ is called as *monomial algebra*. It turns out that the structure of monomial algebras is intimately connected with the automata theory. Next we state the main results in this chapter.

**Results in this thesis**

- We show that the $n \times n$ Permanent (and Determinant) in the quotient algebra $\mathbb{F}\{x_{11}, x_{12}, \ldots, x_{nm}\}/I$ requires $2^{\Omega(n)}$ size ABPs if the ideal $I$ is generated by $2^{o(n)}$ many monomials. Hence, we can extend Nisan’s lower bound argument to noncommutative monomial algebras. Furthermore, the Raz-Shpilka deterministic identity test for noncommutative ABPs [RS05] also carries over to $\mathbb{F}\{X\}/I$.

- In the commutative setting, we prove a $2^{\Omega(n)}$ lower bound for the $n \times n$ Permanent over $\mathbb{Q}\{x_{11}, x_{12}, \ldots, x_{mn}\}/I$, where the monomial ideal $I$ is generated by
o(n/ \log n) monomials. This extends Jerrum and Snir’s [JS82] exponential size lower bound result for monotone arithmetic circuits to a similar lower bound result over commutative monomial algebras.

We also study the Monomial Search Problem. This is a natural search version of polynomial identity testing: Given a polynomial \( f \in \mathbb{F}\{x_1, \ldots, x_n\} \) (or, in the commutative case \( f \in \mathbb{F}[x_1, \ldots, x_n] \)) of total degree \( d \) by an arithmetic circuit or an ABP, the problem is to find a nonzero monomial of the polynomial \( f \). We give a randomized NC\(^2\) algorithm for finding a nonzero monomial and its coefficient in both the commutative as well as the noncommutative setting.

The results presented in this chapter appeared in [AJ09].

## 1.4 Hadamard Product of Polynomials and the Identity Testing Problem

In the work presented in this thesis we introduce and study the Hadamard product of the multivariate polynomials in the free noncommutative polynomial ring \( \mathbb{F}\{x_1, x_2, \cdots, x_n\} \). Our definition of the Hadamard Product can be seen as an algebraic generalization of the intersection of the formal languages. Our definition is motivated by the well know Hadamard product of matrices. Hadamard product of matrices of same dimension is simply the entry-wise product.

Suppose \( X = \{x_1, x_2, \cdots, x_n\} \) is a set of \( n \) noncommuting variables. For a field \( \mathbb{F} \) let \( \mathbb{F}\{x_1, x_2, \cdots, x_n\} \) denote the free noncommutative polynomial ring over \( \mathbb{F} \) generated by the variables in \( X \). We define the Hadamard product of polynomials as follows. Let \( f, g \in \mathbb{F}\{X\} \) where \( X = \{x_1, x_2, \cdots, x_n\} \). The Hadamard product of \( f \) and \( g \), denoted \( f \circ g \), is the polynomial \( f \circ g = \sum_m a_m b_m m \), where \( f = \sum_m a_m m \) and \( g = \sum_m b_m m \), where the sums index over monomials \( m \).

To see the connection of this definition with that of Hadamard product of two matrices we recall the definition of communication matrices [N91] associated with a degree \( d \) homogeneous polynomial \( f \in \mathbb{F}\{X\} \). For \( k = 1, \ldots, d \) the communication matrix \( M_k(f) \)
has its rows indexed by degree $k$ monomials and columns by degree $d - k$ monomials and the $(m, m')^{th}$ entry of $M_k(f)$ is the coefficient of $mm'$ in $f$. It follows easily that Hadamard product of communication matrices associated with two polynomials $f$ and $g$ is same as the communication matrix associated with their Hadamard product (as defined above).

**Results in this thesis**

We explore the arithmetic circuit and the branching program complexity of the Hadamard product of the polynomials when they are individually given by arithmetic circuits and/or algebraic branching programs. We also study the problem of polynomial identity testing for noncommutative ABPs. Using the results on the Hadamard product of polynomials we give a tight classification for the identity testing problem in case of the field of rationals.

We show that the noncommutative branching program complexity of the Hadamard product $f \circ g$ is upper bounded by the product of the branching program sizes for $f$ and $g$. This upper bound is natural because we know from Nisan’s seminal work [N91] that the algebraic branching program (ABP) complexity $B(f)$ is well characterized by the ranks of its “communication” matrices $M_k(f)$, and the rank of Hadamard product $A \circ B$ of two matrices $A$ and $B$ is upper bounded by the product of their ranks. Our proof is constructive: we give a deterministic logspace algorithm for computing an ABP for $f \circ g$.

We apply the above result on the Hadamard Product of two polynomials given by ABPs to tightly classify the identity testing problem for noncommutative ABPs over field of rationals. It is shown by Raz and Shpilka [RS05] that the polynomial identity testing problem for noncommutative ABPs can be solved in deterministic polynomial time. Using result on Hadamard Product of two ABPs, we show that the identity testing problem for noncommutative ABPs over rationals is equivalent to the matrix singularity problem under logspace many-one reductions. Matrix singularity problem is to check whether given integer square matrix is singular or not. It is shown in [AO96] that matrix singularity problem is complete for $C=L$ with respect to logspace many-one reductions when the field is of rational numbers. So our result implies that the identity testing problem in...
case of rationals is $C_{\infty}\text{L}$-complete which is known to be contained in deterministic $\text{NC}^2$.

We show that, the identity testing problem for the noncommutative ABPs over finite field of characteristic $p$ is equivalent to Matrix Singularity problem over field of characteristic $p$ under randomized logspace reduction. Firstly this reduction shows a randomized $\text{NC}^2$ upper bound on the complexity of the problem and it follows from [AO96] that the problem is in randomized $\text{Mod}_p\text{L}$. Using standard amplification techniques we get a $\text{Mod}_p\text{L}/\text{Poly}$ upper bound. We also investigate the parallel complexity of the problem. We show that Raz-Shpilka identity test can be parallelized which gives a $\text{NC}^3$ upper bound for the identity testing problem for the non-commutative ABPs over any field.

It turns out that the problem is hard (with respect to logspace many-one reductions) for both $\text{NL}$ and $\text{Mod}_p\text{L}$. Hence, it is not likely to be easy to improve the upper bound unconditionally to $\text{Mod}_p\text{L}$ (it would imply that $\text{NL}$ is contained in $\text{Mod}_p\text{L}$). Nevertheless it is an interesting question whether we can show deterministic $\text{NC}^2$ upper bound on the identity testing problem for noncommutative ABPs over finite fields?

We explore the expressive power of the Hadamard product of two polynomials when either or both of them given by arithmetic circuit. We show that if either of the two polynomials is given by an ABP the we can efficiently (in logspace) compute an arithmetic circuit for the Hadamard product of the polynomials. But if both the polynomials are given by arithmetic circuits then it is not easy to come up with an efficient algorithm to compute an arithmetic circuit for the Hadamard product of the two polynomials (We show that such an algorithm would imply a non-trivial circuit-size lower bound).

We also consider following identity testing question: Given two polynomials $f, g \in \mathbb{F}\{X\}$ either by an ABP or by an arithmetic circuit check whether $f \circ g$ is identically zero. We show that if both the polynomials are given by arithmetic circuits the problem is coNP-hard even when the circuits are monotone. Whereas, if either of the polynomials is given by an ABP the problem has polynomial time algorithm.

The work presented in this chapter appeared in [AJS09].