MINIMIZING POLYNOMIAL FUNCTIONS ON QUANTUM COMPUTERS

RAOUF DRIDI*, HEDAYAT ALGHASSI† AND SRIDHAR TAYUR‡

This expository paper reviews some of the recent uses of computational algebraic geometry in classical and quantum optimization. The paper assumes an elementary background in algebraic geometry and adiabatic quantum computing (AQC), and concentrates on presenting concrete examples (with Python codes tested on a quantum computer) of applying algebraic geometry constructs: solving binary optimization, factoring, and compiling. Reversing the direction, we also briefly describe a novel use of quantum computers to compute Groebner bases for toric ideals. We also show how Groebner bases play a role in studying AQC at a fundamental level within a Morse theory framework. We close by placing our work in perspective, by situating this leg of the journey, as part of a marvelous intellectual expedition that began with our ancients over 4000 years ago.

1. Introduction

The present paper tells the new story of the growing romance between two protagonists: algebraic geometry and adiabatic quantum computations. An algebraic geometer, who has been introduced to the notion of Ising Hamiltonians, will quickly recognize the attraction in this relationship. However, for many physicists, this connection could be surprising, primarily because of their pre-conception that algebraic geometry is just a very abstract branch of pure mathematics. Although this is somewhat true— that is, algebraic geometry today studies variety of sophisticated objects such as schemes and stacks at heart, those are tools for studying the same problem that our ancients grappled with: solving systems of polynomial equations.

A more known relationship is the one between algebraic geometry and classical polynomial optimization which dates back to the early 90s, with the work of B. Sturmfels and collaborators. The application of algebraic geometry to integer programming can be found in. We take this occasion of an invited paper to introduce both classical and quantum optimization applications of algebraic geometry (the latter, conceived by the authors) through a number of concrete examples, with minimum possible abstraction, with the hope that it will serve as a teaser to join us in this leg of a marvelous expedition that began with the pioneering contributions of the Egyptian, Vedic, and pre-Socrates Greek priesthods.

2. The Profound Interplay Between Algebra and Geometry

In mathematics, there are number of dualities that differentiate it from other sciences. Through these dualities, data transcend abstraction, allowing different interpretations
and access to different probing approaches. One of these is the duality between the category of (affine) algebraic varieties (i.e., zero loci of systems of polynomial equations) and the category of (finitely generated with no nilpotent elements) commutative rings:

\[ \text{affine algebraic varieties} \Rightarrow \text{coordinate rings} \]

(2.1)

Because of this equivalence, we can go back and forth between the two equivalent descriptions, taking advantage of both worlds.

**Example 1** Before we go any deeper, here is an example of an algebraic variety

\[ V := \text{the unit circle in } \mathbb{R}^2. \]

(2.2)

The very same data (set of points at equal distance from the origin) is captured algebraically with the coordinate ring

\[ \mathbb{Q}[x, y]/(x^2 + y^2 - 1) \]

(2.3)

As its name indicates, the coordinate ring provides a coordinate system for the geometrical object \( V \).

We write \( \mathbb{Q}[x_0, ..., x_{n-1}] \) for the ring of polynomials in \( x_0, ..., x_{n-1} \) with rational coefficients (at some places, including the equivalence above, the field of coefficients \( \mathbb{Q} \) should be replaced by its algebraic closure! In practice, this distinction is not problematic and can be safely swept under the rug). Let \( S \) be a set of polynomials \( f \in \mathbb{Q}[x_0, ..., x_{n-1}] \). Let \( \mathcal{V}(S) \) denotes the algebraic variety defined by the polynomials \( f \in S \), that is, the set of common zeros of the equations \( f = 0 \), \( f \in S \). The system \( S \) generates an ideal \( I \) by taking all linear combinations over \( \mathbb{Q}[x_0, ..., x_{n-1}] \) of all polynomials in \( S \); we have \( \mathcal{V}(S) = \mathcal{V}(I) \): The ideal \( I \) reveals the hidden polynomials that are the consequence of the generating polynomials in \( S \). For instance, if one of the hidden polynomials is the constant polynomial \( 1 \) (i.e., \( 1 \in I \)), then the system \( S \) is inconsistent (because \( 1 \neq 0 \)). To be precise, the set of all hidden polynomials is given by the so-called radical ideal \( \sqrt{I} \), which is defined by

\[ \sqrt{I} = \left\{ g \in \mathbb{Q}[x, ..., x_n] \mid \exists r \in \mathbb{N} : g^r \in I \right\}. \]

We have:

**Proposition 1** \( I(\mathcal{V}(I)) = \sqrt{I} \)

Of course, the radical ideal \( \sqrt{I} \) is infinite. Luckily, thanks to a prominent technical result (i.e., Dickson’s lemma), it has a finite generating set i.e., a Groebner basis \( \mathcal{B} \), which one might take to be a triangulization of the ideal \( \sqrt{I} \). In fact, the computation of Groebner bases generalizes Gaussian elimination in linear systems.

**Proposition 2** \( \mathcal{V}(S) = \mathcal{V}(I) = \mathcal{V}(\sqrt{I}) = \mathcal{V}(\mathcal{B}) \)

Instead of giving the technical definition of what a Groebner basis is (which can be found in other books) let us give an example (for simplicity, we use the term “Groebner bases” to refer to reduced Groebner bases, which is, technically what we are working with):

**Example 2** Consider the system by

\[ S = \{ x^2 + y^2 + z^2 - 4, x^2 + 2y^2 - 5, xz - 1 \}. \]

We want to solve \( S \). One way to do so is to compute a Groebner basis for \( S \). In Figure 1, the output of the cell number 4 gives a Groebner basis of \( S \). We can see that the initial system has been triangulized: The last equation contains only the variable \( z \), whilst the second has an additional variable, and so on. The variable \( z \) is said to be eliminated with respect to the rest of the variables. When computing the Groebner basis, the underlying algorithm (Buchberger’s algorithm) uses the ordering \( x > y > z \) (called lexicographical ordering) for the computing of two internal calculations: crossmultiplications and Euclidean divisions. The program tries to isolate \( z \) first, then \( z \) and \( y \), and finally \( x \); \( y \), and \( z \) (all variables). It is clear that different orderings yield different Groebner bases.

The mathematical power of Groebner bases doesn’t stop at solving systems of algebraic equations. The applicability of Groebner bases goes well beyond this: it gives necessary and sufficient conditions for the existence of solutions. Let us illustrate this with an example.
Example 3 Consider the following 0-1 feasibility problem
\[
\begin{align*}
    x_1 + x_3 &= b_1, \\
    x_1 + ax_2 &= b_2,
\end{align*}
\]
(2.4)

with \( x_i \in \{0,1\} \) for \( i = 1, 2, 3 \). By putting the variables \( a_2, b_1, \) and \( b_2 \) to the rightmost of the ordering, we obtain the set of all \( a_2, b_1, \) and \( b_2 \) for which the system is feasible.

The notebook in Figure 2 shows the details of the calculations as well as the conditions on the variables \( a_2, b_1, \) and \( b_2 \).

This machinery can be put in more precise wording as follows:

Theorem 1 Let \( I \subset \mathbb{Q}[x_0, \ldots, x_{n-1}] \) be an ideal, and let \( B \) be a reduced Groebner basis of \( I \) with respect to the lex order \( x_0 \succ \ldots \succ x_{n-1} \). Then, for every \( 0 \leq i \leq n-1 \), the set
\[
\mathcal{B} \cap \mathcal{Q}[x_0, \ldots, x_{n-1}]
\]
is a Groebner basis of the ideal \( I \cap \mathcal{Q}[x_0, \ldots, x_{n-1}] \).

As previously mentioned, this elimination theorem is used to obtain the complete set of conditions on the variables \( x_i, \ldots, x_{n-1} \), such that the ideal \( I \) is not empty. For instance, if the ideal represents a system of algebraic equations and these equations are (algebraically) dependent on certain parameters, then the intersection (2.5) gives all necessary and sufficient conditions for the existence of solutions.

3. The Innate Role of Algebraic Geometry in Binary Optimization

By now, it should not be surprising to see algebraic geometry emerges when optimizing polynomial functions. Here, we expand on this with two examples of how algebraic geometry solves the binary polynomial optimization

\[
(P) : \arg\min_{y_0, \ldots, y_{m-1}} f(y_0, \ldots, y_{m-1}) \quad (3.1)
\]

The first method we review here was introduced in\(^{10}\) (different from another previous method that is studied in\(^{8}\), which we discuss in a later section). The second method we review here is new, and is an adaptation of the method described in\(^{6}\) to the binary case.

3.1 A general method for solving binary optimizations: The key idea is to consider the ideal

\[
I = \left\{ z - f(y_0, \ldots, y_{m-1}) \right\},
\]

\[
y_0^2 - y_0, \ldots, y_{m-1}^2 - y_{m-1},
\]
where we note the appearance of the variable \( z \). This new variable covers the range of the function \( f \). Consequently, if we compute a Groebner basis with an elimination ordering in which \( z \) appears at the
rightmost, we obtain a polynomial in \( z \) that gives all values of \( f \). Take, then, the smallest of those values and substitute in the rest of the basis and solve.

**Example 4** Consider the following problem

\[
\begin{align*}
\arg\min_{y \in (0,1)} & \quad y_1 + 2y_2 + 3y_3 + 3y_4, \\
\quad & \quad y_1 + y_2 + 2y_3 + y_4 = 3.
\end{align*}
\] (3.2)

Figure 3 details the solution.

The variety \( \mathcal{V}(\check{I}) \) gives the set of local minima of the function \( f \). Its coordinate ring is the residue algebra

\[
A := \mathbb{Q}[y_0, \ldots, y_{m-1}, \alpha_0, \ldots, \alpha_{m-1}] / \check{I}
\] (3.5)

Let us define the linear map

\[
m_f : A \rightarrow A, \quad g \mapsto \tilde{f}g
\] (3.6)

Because the number of local minima is finite, the residue algebra \( A \) is finite dimensional. Because of this, the following is true

- The values of \( \tilde{f} \), on the set of critical points \( \mathcal{V}(\check{I}) \), are given by the eigenvalues of the matrix \( m_f \).

- The eigenvalues of \( m_{y_i} \) and \( m_{\alpha_i} \) give the coordinates of the points of \( \mathcal{V}(\check{I}) \).

- If \( v \) is an eigenvector for \( m_f \), then it is also an eigenvector for \( m_{y_i} \) and \( m_{\alpha_i} \) for \( 0 \leq i \leq m - 1 \).

We need to compute a basis for \( A \). This is done by first computing a Groebner basis for \( \check{I} \) and then extracting the standard monomials (i.e., the monomials in \( \mathbb{Q}[y_0, \ldots, y_{m-1}, \alpha_0, \ldots, \alpha_{m-1}] \) that are not divisible by the leading term of any element in the Groebner basis). In the simple example below, we do not need to compute any Groebner basis, because \( \check{I} \) is a Groebner basis with respect to \( \text{plex}(\alpha, y) \).

**Example 5** We illustrate this on

\[
f = 2 + 7x_4 + 2x_5 + 2x_4x_3 - 2x_3x_2 - x_1 - 4x_4x_1 - 2x_3x_1 + x_2x_1
\]

where \( x_i \in \{0, 1\} \). A basis for the residue algebra \( A \) is given by the set of the 16 monomials

\[
\{1, x_4, x_3, x_4x_3, x_2, x_4x_2, x_3x_2, x_3x_2x_4, x_1, x_4x_1, x_3x_1, x_1x_3, x_2x_1, x_4x_1x_3, x_3x_2x_4, x_1x_3x_2, x_1x_3x_2x_4\}.
\]
The matrix $m_j$ is
\[
\begin{bmatrix}
2 & 7 & 2 & 0 & 0 & -2 & 0 & -1 & -4 & -2 & 0 & 1 & 0 & 0 & 0 \\
0 & 9 & 0 & 4 & 0 & 0 & 0 & -2 & 0 & -5 & 0 & -2 & 0 & 1 & 0 \\
0 & 0 & 4 & 9 & 0 & 0 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 1 \\
0 & 0 & 0 & 13 & 0 & 0 & -2 & 0 & 0 & 0 & -7 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 2 & 7 & 0 & 2 & 0 & 0 & 0 & 0 & -4 & -2 & 0 \\
0 & 0 & 0 & 0 & 0 & 9 & 0 & 2 & 0 & 0 & 0 & -4 & 0 & -2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 9 & 0 & 0 & 0 & 0 & -2 & -4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 & -6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 2 & 1 & 0 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 2 & 0 & 1 & 0 & -2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & -2 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\]

We obtain the following eigenvalues for $m_j$ :
\[
\{ 0, 1, 2, 4, 5, 6, 9, 11, 13 \}.
\]

This is also the set of values that $f$ takes on $\mathcal{V}(\hat{I})$.

The eigenvector $v$ that corresponds to the eigenvalue 0 is the column vector
\[
v := (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0)^T.
\]

This eigenvector is used to find the coordinates of $\hat{x} \in \mathcal{V}(\hat{I})$ that minimizes $f$. The coordinates of the global minimum $\hat{x} = (\hat{x}_0, ..., \hat{x}_{m-1})$ are defined by $m_j v = \hat{x} v$, and this gives $x_1 = x_2 = x_3 = 1, x_4 = 0, \alpha_1 = 2, \alpha_2 = \alpha_3 = 2, \alpha_4 = 5$.

4. Factoring on Quantum Annealers

This section reviews the use of the Groebner bases machinery in the factoring problem on current quantum annealers (introduced in\textsuperscript{13}). We need to deal with three key constraints: first, the number of available qubits, second, the limited dynamic range for the allowed values of the couplers (i.e., coefficients of the quadratic monomials in the cost function), and third, the sparsity of the hardware graph.

4.1 Reduction : In general, reducing a polynomial function $f$ into a quadratic function necessitates the injection of extra variables (the minimum reduction is given in terms of toric ideals\textsuperscript{14}). However, in certain cases, the reduction to QUBOs can be done without the additional variables. This is the example of the Hamiltonian that results from the long multiplication\textsuperscript{13}. In fact, in addition to reduction, we can also adjust the coefficients to be within the dynamic range needed, at the same time. Consider the quadratic polynomial
\[
H_{ij} := Q_j P_j + S_{ij} + Z_{ij} - S_{i+1,j-1} - 2Z_{ij+1},
\]

with the binary variables $P_j, Q_j, S_{ij}, Z_{ij}, Z_{ij+1}, Z_{i+1,j-1}$. The goal is to solve $H_{ij}$ (obtain its zeros) by converting it into a QUBO. Instead of directly squaring the function $H_{ij}$ (naive approach) and then reducing the cubic function result into a quadratic function by adding extra variables, we compute a Groebner basis $\mathcal{B}$ of the system
\[
S = \{ H_{ij} \} \cup \{ x^2 - x, x \in \{ P_j, Q_j, S_{i,j}, S_{i+1,j-1}, Z_{i,j}, Z_{i,j+1} \} \},
\]

and look for a positive quadratic polynomial $H_{ij}^+ = \sum_{i \in \mathbb{B} \text{ deg}(t) \leq 2} q_i t^i$ in the ideal generated by $S$. Note that global minima of $H_{ij}^+$ are the zeros of $H_{ij}$.

The Groebner basis $\mathcal{B}$ is
\[
\begin{align*}
t_1 & := Q_j P_j + S_{ij} + Z_{ij} - S_{i+1,j-1} - 2Z_{ij+1}, \\
t_2 & := (Z_{ij+1} + Z_{ij}) S_{i+1,j-1} + (Z_{ij+1} - 1)Z_{ij}, \\
t_3 & := (S_{i+1,j-1} + Z_{ij+1} + Z_{ij} - S_{ij}) S_{ij} + Z_{ij+1} - Z_{ij}, \\
t_4 & := (S_{i+1,j-1} + Z_{ij+1} + Z_{ij}) S_{ij} - S_{ij} + Z_{ij+1} - Z_{ij}, \\
t_5 & := (S_{i+1,j-1} + Z_{ij+1} + S_{ij}) Q_j - S_{ij} + Z_{ij} + S_{ij+1,j-1}, \\
t_6 & := (S_{i+1,j-1} + Z_{ij+1} + S_{ij}) P_j - S_{ij} + Z_{ij} + S_{ij+1,j-1} + 2Z_{ij+1},
\end{align*}
\]

in addition to 3 more cubic polynomials.

We take $H_{ij}^+ = \sum_{i \in \mathbb{B} \text{ deg}(t) \leq 2} q_i t^i$; and solve for the $a_i$. We can require that the coefficients $a_i$ are subject to the dynamic range allowed by the quantum processor (e.g., the absolute values of the coefficients of $H_{ij}^+$, with respect to the variables $P_j, Q_j, S_{ij}, S_{i+1,j-1}, Z_{ij}$, and $Z_{ij+1}$, be within $[1-\varepsilon, 1+\varepsilon]$). The ensemble of these constraints translates
4.2 Embedding: The connectivity graph of the resulting quadratic polynomial \( H^+_{ij} \) is the complete graph \( K_6 \). Although embedding this into current architectures is not evident, the situation becomes better with upcoming architectures (e.g., D-Wave’s next generation quantum processors\(^{15}\)).

5. Compiling on Quantum Annealers

Compiling the problem \((P)\) in AQC, consists of two steps: reduction of the problem’s polynomial function into a quadratic function (covered above) and later embedding the graph of the quadratic function inside the quantum annealer’s hardware graph. This process can be fully automatized using the language of algebraic geometry\(^{14}\). We review here the key points of this automatization, through a simple example.

Let us first explain what is meant by embeddings (and introduce the subtleties that come with). Consider the following optimization problem that we wish to solve on the D-Wave 2000Q quantum processor:

\[
(P): \arg \min_{y_0, \ldots, y_9} y_0 \sum_{i=1}^{8} c_i y_i. \tag{5.1}
\]

Before we start annealing, we need to map the logical variables \( y_i \) to the physical qubits of the hardware. Similarly, the quadratic term \( c_{y_0 y_j} \) needs to be mapped into a coupling between physical qubits with strength given by the coefficient \( c_j \). Not surprisingly, this mapping can not always be a simple matching—because of the sparsity of the hardware graph (Chimera in our case). This is true for our simple example; the degree of the central node is 8, so a direct matching inside Chimera, where the maximum degree is 6, is not feasible. Thus, we stretch the definition of embedding. Instead, we allow nodes to elongate or, as an algebraic geometer will say, to blow up. In particular, if we blow up the central node \( y_0 \) into an edge, say the edge \((x_5, x_9)\), we can then redistribute the surrounding nodes \( y_1, \ldots, y_8 \), at these two duplicates of \( y_0 \). In general, one needs a sequence of blow ups, which turns out to be a hard problem. What makes the problem even harder is that not all embeddings are equally valued. It is important to choose embeddings that have, among others, smaller chains, as illustrated in Figure 5. Of course, this is in addition to minimizing the overall number of physical qubits used.

5.1 Embeddings as fiber-bundles: One way to think about embedding the logical graph \( Y \) into the hardware graph \( X \) is in terms of fiber-bundles. This *equational* formulation makes the connection with algebraic geometry. The general form of such fiber-bundles is

\[
\pi(x_j) = \sum_{ij} \alpha_{ij} y_j \tag{5.2}
\]

with

\[
\sum_{ij} \alpha_{ij} = \beta_j, \alpha_{ij} = 0, \alpha_{ij}(\alpha_{ij} - 1) = 0,
\]

where the binary number \( \beta_j \) is 1 if the physical qubits \( x_j \) is used and 0 otherwise. We write \( \text{domain}(\pi) = \text{Vertices}(X) \) and \( \text{support}(\pi) = \text{Vertices}(X^\beta) \subset \text{subgraph}(X) \). The fiber of the map \( \pi \) at \( y_j \in \text{Vertices}(Y) \) is given by

\[
\pi^{-1}(y_j) = \phi(y_j) = \{ x_i \in \text{Vertices}(X) | \alpha_{ij} = 1 \} \tag{5.3}
\]

The conditions on the parameters \( \alpha_{ij} \) guarantee that fibers don’t intersect (i.e., \( \pi \) is well defined map). In addition to these conditions, two more conditions need to be satisfied: (i) Pullback Condition: the logical graph \( Y \) embeds entirely inside \( X \) (ii) Connected Fiber Condition: each fiber is a connected subgraph (of \( X \)). We will

Figure 4: (Left) The logical graph of the objective function in \((P)\), can not be embedded inside Chimera graph. (Center) We blow up the central node into edges \((x_5, x_9)\) and redistribute the surrounding nodes. (Right) Embedding inside an actual D-Wave 2000Q quantum annealer; in red, the chain of qubits representing the logical qubit \( y_0 \). The missing qubits are faulty.

Figure 5: The depicted embedding (for the problem \((P)\)) has two long chains that don’t persist through the adiabatic evolution (in D-Wave 2000Q processor). In this case, the quantum processor fails to return an answer.

...
not go into the details of these conditions, which can be found in\cite{14}. We illustrate this in a simple example.

**Example 6** Consider the two graphs in Figure 6. In this case, equations (5.2) are given by

\[
\begin{align*}
\alpha_{1,1} \alpha_{1,2} \alpha_{1,1} \alpha_{1,3} & , \\
\alpha_{2,1} \alpha_{2,2} \alpha_{2,3} & , \\
\alpha_{3,1} \alpha_{3,2} \alpha_{3,3} & , \\
\alpha_{4,1} \alpha_{4,2} & , \\
\alpha_{5,1} \alpha_{5,2} & ,
\end{align*}
\]

\begin{align}
\alpha_{1,1} \alpha_{1,2} \alpha_{1,3} & , \\
\alpha_{2,1} \alpha_{2,2} \alpha_{2,3} & , \\
\alpha_{3,1} \alpha_{3,2} \alpha_{3,3} & , \\
\alpha_{4,1} \alpha_{4,2} & , \\
\alpha_{5,1} \alpha_{5,2} & ,
\end{align}

and

\[
\begin{align*}
\alpha_{1,1} + \alpha_{1,2} + \alpha_{1,3} - \beta_1, \\
\alpha_{2,1} + \alpha_{2,2} + \alpha_{2,3} - \beta_2, \\
\alpha_{3,1} + \alpha_{3,2} + \alpha_{3,3} - \beta_3, \\
\alpha_{4,1} + \alpha_{4,2} + \alpha_{4,3} - \beta_4, \\
\alpha_{5,1} + \alpha_{5,2} + \alpha_{5,3} - \beta_5,
\end{align*}
\]

![Figure 6: The set of all fiber bundles \( \pi : X \rightarrow Y \) defines an algebraic variety. This variety is given by the Groebner basis (5.9).](image)

The Pullback Condition reads

\[
\begin{align*}
-1 + \alpha_{4,1} \alpha_{5,1} + \alpha_{3,1} \alpha_{4,2} + \alpha_{1,1} \alpha_{2,2} + \alpha_{3,2} \alpha_{4,1} + \alpha_{1,2} \alpha_{2,1} \\
+ \alpha_{1,2} \alpha_{4,1} + \alpha_{2,2} \alpha_{3,1} + \alpha_{1,1} \alpha_{4,2} + \alpha_{2,1} \alpha_{3,2} + \alpha_{4,2} \alpha_{5,1}, \\
-1 + \alpha_{3,3} \alpha_{4,1} + \alpha_{1,3} \alpha_{2,1} + \alpha_{2,3} \alpha_{3,1} + \alpha_{4,1} \alpha_{5,3} + \alpha_{1,3} \alpha_{4,1} + \alpha_{1,1} \alpha_{2,3} + \alpha_{4,3} \alpha_{5,1} + \alpha_{2,1} \alpha_{3,3} + \alpha_{3,1} \alpha_{4,3} + \alpha_{1,1} \alpha_{4,3}, \\
-1 + \alpha_{3,3} \alpha_{4,2} + \alpha_{1,2} \alpha_{2,3} + \alpha_{1,2} \alpha_{4,3} + \alpha_{1,3} \alpha_{2,2} + \alpha_{1,3} \alpha_{4,2} + \alpha_{2,3} \alpha_{3,3} + \alpha_{4,2} \alpha_{5,2} + \alpha_{2,2} \alpha_{3,3} + \alpha_{4,2} \alpha_{5,2},
\end{align*}
\]

Finally, the Connected Fiber Condition is given by

\[
\begin{align*}
-\alpha_{1,1} \alpha_{2,1} \alpha_{5,1} - \alpha_{1,1} \alpha_{3,1} \alpha_{5,1} - \alpha_{1,2} \alpha_{2,2} \alpha_{5,2}, \\
-\alpha_{1,2} \alpha_{3,2} \alpha_{5,2} - \alpha_{1,3} \alpha_{2,3} \alpha_{5,3} - \alpha_{1,3} \alpha_{3,3} \alpha_{5,3}
\end{align*}
\]

We can then use the elimination theorem to obtain all embeddings of \( Y \) inside \( X \) (by putting the variables \( \beta_i \) to the right most of the elimination order). A part of the Groebner basis is given by

\[
B = \{ \beta_1 - 1, \beta_2 - 1, \beta_3 - 1, \beta_4 - 1, \beta_5^2 - \beta_5, \alpha_1^2 - \alpha_4 \beta_1 \}
\]

In particular, the intersection \( B \cap Q[\beta] = \{ \beta_1 - 1, \beta_2 - 1, \beta_3 - 1, \beta_4 - 1, \beta_5^2 - \beta_5 \} \) gives the two \( Y \) minors (i.e., subgraphs \( X^\beta \)) inside \( X \). The remainder of \( B \) gives the explicit expressions of the corresponding mappings.

**5.2 Symmetry Reduction** : Many of the embeddings acquired using the above method, are redundant. We can eliminate this redundancy in a mathematically elegant way using the theory of invariants [Olv99] (on top of the algebraic geometrical formulation). First, we fold the hardware graph along its symmetries and then proceed as before. This amounts to re-expressing the quadratic form of the hardware graph in terms of the invariants of the symmetry.

**Example 7** Continuing with the same example: The quadratic form of \( X \) is:

\[
Q_X(x) = x_1^2 + x_2 x_3 + x_3 x_4 + x_4 x_5 + x_5^2 (5.10)
\]

Exchanging the two nodes \( x_1 \) and \( x_3 \) is a symmetry for \( X \); and the quantities \( K = x_1 + x_3, x_2, x_4, \) and \( x_5 \) are invariants of this symmetry. In terms of these invariants, the quadratic function \( Q_X(x) \), takes the simplified form:

\[
Q_X(x, K) = K x_2 + K x_4 + x_5^2 (5.11)
\]

which shows (as expected) that graph \( X \) can be folded into...
a chain (given by the new nodes \([x_2, K, x_4, x_5]\)). The surjective homomorphism \(\pi : X \to Y\) now takes the form

\[
K = a_{01}v_1 + a_{02}v_2 + a_{03}v_3.
\]

\(x_i = a_{i1}v_1 + a_{i2}v_2 + a_{i3}v_3\) for \(i = 2, 4, 5\).

The table below compares the computations of the surjections \(\pi\) with and without the use of invariants:

<table>
<thead>
<tr>
<th></th>
<th>original coords</th>
<th>invt coords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time for computing a Groebner basis (in secs)</td>
<td>0.122</td>
<td>0.039</td>
</tr>
<tr>
<td>Number of defining equations</td>
<td>58</td>
<td>30</td>
</tr>
<tr>
<td>Maximum degree in the defining eqns</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Number of variables in the defining eqns</td>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>Number of solutions</td>
<td>48</td>
<td>24</td>
</tr>
</tbody>
</table>

In particular, the number of solutions is down to 24, that is, four (non symmetric) minors times the six symmetries of the logical graph \(Y\).

### 6. Quantum Computing for Algebraic Geometry

Here we give an example that goes in the opposite direction of what we have covered so far. We show how quantum computers can be used to compute algebraic geometrical structures that are exponentially hard to compute classically. Our attention is directed to a prominent type of polynomial ideals; the so-called toric ideals and their Groebner bases. In the context of the theory of integer optimization, this gives a novel quantum algorithm for solving IP problems (a quantum version of Conti and Traverso algorithm, QAOA\(^{18}\)) can be used to compute Groebner bases for the toric ideal \(J_d\). The example we choose is taken from Chapter 8. The matrix \(A\) is given by

\[
A = \begin{pmatrix} 4 & 5 & 1 & 0 \\ 2 & 3 & 0 & 1 \end{pmatrix}
\]

The kernel is easily obtained (with polynomial complexity). It is the two dimensional \(\mathbb{Z}\)–vector space spanned with \(((1, 0, -4, -2), (0, 1, -5, -3))\): We define \(u = u_+ - u_- \in \mathbb{Z}^+ \oplus \mathbb{Z}^+\) runs over all integer vectors in \(\text{Ker}_Z A\), the kernel of the matrix \(A\). It is not hard to see that the elimination theorem that we have used repeatedly can also be used here to compute a Groebner basis for the toric ideal \(J_d\).

Now we explain how AQC (or any quantum optimizer such as Quantum Approximate Optimization Algorithm, QAOA\(^{18}\)) can be used to compute Groebner bases for the toric ideal \(J_d\). The example we choose is taken from Chapter 8. The matrix \(A\) is given by

\[
A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}
\]

The cost function is given by the square of the Euclidean norm of the vector \(Mut\). Figure 7 details the solution of this optimization problem on D-Wave 2000Q quantum processor. Each solution has twelve entries, and is of the form \([a_{0,1}, a_{0,2}, ..., b_{2,1}, b_{2,2}]\), corresponding to the binary decomposition of the integers \(a = \sum_{i=0,1,2} (a_{i,+} - a_{i,-}) 2^i\) and \(b = \sum_{i=0,1,2} (b_{i,+} - b_{i,-}) 2^i\). Qubits marked -1 are not used, so they should be considered equal to zero. The collection of all these solutions translates into the sought Groebner basis.

\[a_n\]

is identified with a Laurent monomial \(y^{|a_n|} = y_1^{a_1} \cdots y_m^{a_m}\). In this case, the toric ideal \(J_d\) associated with the configuration \(A\) is the kernel of the algebra homomorphism

\[
\mathbb{Q}[x] \to \mathbb{Q}[y]
\]

\[x_i \to y^{|a_i|}.
\]

From this it follows that the toric ideal \(J_d\) is generated by the binomials \(x_{u,+} - x_{u,-}\); where the vector \(u = u_+ - u_- \in \mathbb{Z}^+ \oplus \mathbb{Z}^+\) runs over all integer vectors in \(\text{Ker}_Z A\), the kernel of the matrix \(A\). It is not hard to see that the elimination theorem that we have used repeatedly can also be used here to compute a Groebner basis for the toric ideal \(J_d\).

Toric ideals are ideals generated by differences of monomials. Because of this, their Groebner bases enjoy a clear structure given by kernels of integer matrices. Specifically, let \(A = (a_1, ..., a_n)\) be any integer \(m \times n\)-matrix (\(A\) is called configuration matrix). Each column \(a_i = (a_{i1}, ..., a_{im})\)}
The role of the so-called anti-crossings in AQC is well understood. This is expressed as the total adiabatic evolution time being inversely proportional to the square of the minimum energy difference between the two lowest energies of the given Hamiltonian. This minimum is attained at anti-crossings. In this last section, we connect anti-crossings to the theory of Groebner bases (through a quick detour to Morse theory\textsuperscript{20, 21}).

Consider the time dependant Hamiltonian:

\[ H(s) = (1 - s)H_{\text{initial}} + sH_{\text{final}} \]  

(7.1)

To the Hamiltonian (7.1), we assign the function \( f \) given by the characteristics polynomial:

\[ f(s, \lambda) = \det(H(s) - \lambda I). \]  

(7.2)

where \( I \) is the identity \( n \times n \) matrix. The important role that the function \( f \) plays in AQC is described in\textsuperscript{22, 23}. In particular, anti-crossings are now mapped into saddle points of the function \( f \). This is the starting point of the connection with Morse theory, which is explored in details in\textsuperscript{22, 23}.

Here we explain how anti-crossings can be described using Groebner bases. The key fact is that the function \( f \) is a polynomial function of \( s \) and \( \lambda \), and so is any partial derivative of \( f \). Recall that a critical point \( p \) of \( f \) is a point at which the differential map \( df \) is the zero map—that is,

\[ \begin{pmatrix}
   w_2^3 w_3 - w_4^3 \\
   w_1^4 w_4 - w_2^3 w_3 \\
   w_2 w_4 w_3^3 - w_2^3 w_1 \\
   w_3 w_4^3 - w_4^3 w_1
\end{pmatrix} = 0. \]  

(6.5)
the gradient of \( f \) vanishes at \( p \). A critical point is said to be non degenerate (e.g., a saddle point) if the determinant of the Hessian of \( f \) at \( p \) is not zero. Define the ideal \( I \) generated by the two polynomials \( \partial_1 f \) and \( \partial_2 f \). It is clear that the variety of \( I \) gives the set of all critical points of \( f \). To capture the non degeneracy, we need to saturate the ideal \( I \) with the polynomial \( \det(Hessian(f)) \). This saturation is the ideal given by all polynomials in \( I \) that vanish for all the zeros of \( I \) that are not zeros of \( \det(Hessian(f)) \). In other words, a point \( p \) is a non degenerate critical point of the function \( f \) if and only if the remainder NormalForm\(_4\)(\( \det(Hessian(f)) \)) is not zero at \( p \), where \( B \) is a Groebner basis for the ideal \( I \).

8. Summary and Discussion

As we mentioned in the Introduction, we are travellers in a journey that our ancients started. Evidence of “practical mathematics” during 2200 BCE in the Indus Valley has been unearthed that indicates proficiency in geometry. Similarly, in Egypt (around 2000 BCE) and Babylon (1900 BCE), there is good evidence (through the Rhind Papyrus and clay tablets, respectively) of capabilities in geometry and algebra. After the fall of the Indus Valley Civilization (around 1900 BCE), the Vedic period was especially fertile for mathematics, and around 600 BCE, there is evidence that magnetism (discovered near Varanasi) was already used for practical purposes in medicine (like pulling arrows out of warriors injured in battle), as written in Sushruta Brahmana. Magnetism was also independently discovered by pre-Socrates Greeks, as evidenced by the writings of Thales (624-548 BCE), who, along with Pythagoras (570-495 BCE), was also quite competent in geometry. Indeed, well before Alexander (The Great), and the high points of Hellenistic Greek period, there is evidence that the Greeks were already doing some type of algebraic geometry.

Algebra, which is derived from the Arabic word meaning completion or “reunion of broken parts”, reached a new high watermark during the golden age of Islamic mathematics around 10th Century AD. For example, Omar Khayyam (of the Rubaiyat fame) solved cubic equations. The next significant leap in algebraic geometry, a Renaissance, in the 16th and 17th century, is quintessentially European: Cardano, Fontana, Pascal, Descartes, Fermat. The 19th and 20th Century welcomed the dazzling contributions of Laguerre, Cayley, Reimann, Hilbert, Macaulay, and the Italian school led by Castelnuovo, del Pezzo, Enriques, Fano, and Severio. Modern algebraic geometry has been indelibly altered by van der Waerden, Zariski, Weil, and in 1950s and 1960s, by Serre and Grothendieck. Computational algebraic geometry begins with the Buchberger in 1965 who introduced Groebner bases (the first conference on computational algebraic geometry was in 1979).

Magnetism simply could not be explained by classical physics, and had to wait for quantum mechanics. The workhorse to study it mathematically is the Ising model, conceived in 1925. Quantum computing was first introduced by Feynman in 1981. The study of Ising models that formed a basis of physical realization of a quantum annealer (like D-Wave devices) can be traced to the 1989 paper by Ray, Chakrabarti and Chakrabarti. Building on various adiabatic theorems of the early quantum mechanics and complexity theory, adiabatic quantum computing was proposed by Farhi et al in 2001.

Which brings us to current times. The use of computational algebraic geometry (along with Morse homology, Cerf theory and Gauss-Bonnet theorem from differential geometry) in the study of adiabatic quantum computing, and numerically testing our ideas on D-Wave quantum processors, which is a physical realization of an Ising model, is conceived by us, the authors, of this expository article. Let us close with the Roman poet Ovid (43 BC-17 AD): “Let others praise ancient times; I am glad I was born in these.”

References


