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[1] and adiabatic quantum computations[2,3]. An algebraic geometer, who has been introduced to the notion of Ising Hamiltonians[4], 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.

Quantum Computing Group, Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213

* rdridi@andrew.cmu.edu

† halghassi@cmu.edu

‡ stayur@cmu.edu

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[5,6]. The application of algebraic geometry to integer programming can be found in[7,8,9,10]. 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 priesthoods.

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:

$$\{\text{affine algebraic varieties}\} \simeq \{\text{coordinate rings}\}^{op} \tag{2.1}$$

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*

$$\mathcal{V} := \textit{the unit circle in } \mathbb{R}^2. \tag{2.2}$$

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

$$\mathbb{Q}[x, y]\Big/\left\langle x^2 + y^2 - 1\right\rangle \text{ = polynomials, in } x \text{ and } y; \text{ mod } (x^2 + y^2 - 1): \tag{2.3}$$

As its name indicates, the coordinate ring provides a coordinate system for the geometrical object $\mathcal{V}$.

We write $\mathbb{Q}[x_0, \ldots, x_{n-1}]$ for the ring of polynomials in $x_0, \ldots, x_{n-1}$ 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 $f \in \mathbb{Q}[x_0, \ldots, x_{n-1}]$. 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 $\mathbb{Q}[x_0, \ldots, x_{n-1}]$ 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 $\sqrt{I} = \left\{ g \in \mathbb{Q}[x_1, \ldots, x_n] \,\middle|\, \exists r \in \mathbb{N} : g^r \in I \right\}$. We have:

Proposition 1 $I\left(\mathcal{V}(I)\right) = \sqrt{I}$

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 triangularization 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[1] and in many other text 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 lexigraphical 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.

```
In [1]: from sympy import *

        x, y, z = symbols('x y z')

In [2]: eqs = [x**2+y**2+z**2-4, x**2+2*y**2-5, x*z-1]

In [3]: # to compute a Groebner basis we need a "monomial order" = ordering relation on the variables x, y and z

        result = groebner(eqs,  x,y, z,  order='lex')

In [4]: for eq in list(result):
            print eq
            print "\n"

        x + 2*z**3 - 3*z


        y**2 - z**2 - 1


        2*z**4 - 3*z**2 + 1
```

Figure 1: Jupyter notebook for computing Groebner bases using Python package sympy. More efficient algorithms exist (e.g.,[11,12]).

Example 3 *Consider the following 0-1 feasibility problem*

$$\begin{cases} x_1 + x_1 + x_3 = b_1, \\ x_1 + a_2 x_2 \quad = b_2, \end{cases} \tag{2.4}$$

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 $\mathcal{I} \subset \mathbb{Q}\left[x_0, \ldots x_{n-1}\right]$ be an ideal, and let $\mathcal{B}$ be a reduced Groebber basis of $\mathcal{I}$ with respect to the lex order $x_0 \succ \ldots \succ x_{n-1}$. Then, for every $0 \le l \le n-1$, the set

$$\mathcal{B} \cap \mathbb{Q}\left[x_0, \ldots x_{n-1}\right] \tag{2.5}$$

is a Groebner basis of the ideal $\mathcal{I} \cap \mathbb{Q}\left[x_0, \ldots x_{n-1}\right]$.

As previously mentioned, this elimination theorem is used to obtain the complete set of conditions on the variables x_l, ... , x_{n-1}, such that the ideal $\mathcal{I}$ 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

$$(P): argmin_{(y_0,\ldots,y_{m-1}) \in \{0,1\}^m} f\left(y_0, \ldots, y_{m-1}\right) \tag{3.1}$$

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

$$I = \left\langle z - f\left(y_0, \ldots, y_{m-1}\right), \right.$$

$$\left. y_0^2 - y_0, \ldots, y_{m-1}^2 - y_{m-1} \right\rangle,$$

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

```
In [1]: from sympy import *

        x1, x2, x3, a2, b1, b2 = symbols('x1, x2, x3,  a2, b1, b2')

In [2]: eqs = [x1+x2+x3-b1, x1+a2*x2-b2]
        eqs = eqs + [x*(x-1) for x in [x1, x2, x3]]
        print(eqs)

        [-b1 + x1 + x2 + x3, a2*x2 - b2 + x1, x1*(x1 - 1), x2*(x2 - 1), x3*(x3 - 1)]

In [3]: # Computing a Groebner basis
        B = groebner(eqs,  x1, x2, x3,   a2, b1, b2,   order='lex')

In [4]: # The intersection:  B ∩ Q[a2, b1, b2]
        for eq in list(B)[10:]:
            print str(eq) + ","

        a2**2*b1**2 - 2*a2**2*b1*b2 - a2**2*b1 + 2*a2**2*b2 + a2*b1**2 - 2*a2*b1*b2 - a2*b1 + 4*a2*b2**2 - 2*a2*b2 - 3*b1**2*
        b2**2 + 3*b1**2*b2 + 2*b1*b2**3 + 9*b1*b2**2 - 11*b1*b2 - 6*b2**3 - 2*b2**2 + 8*b2,
        a2**2*b2**2 - a2**2*b2 - 2*a2*b2**3 + 3*a2*b2**2 - a2*b2 + b2**4 - 2*b2**3 + b2**2,
        a2*b1**3 - 3*a2*b1**2 + 2*a2*b1 + b1**3 - 3*b1**2*b2 - 3*b1**2 + 9*b1*b2 + 2*b1 - 6*b2,
        a2*b1**2*b2 - 3*a2*b1*b2 + 2*a2*b2 - b1**2*b2**2 + b1**2*b2 + 3*b1*b2**2 - 3*b1*b2 - 2*b2**2 + 2*b2,
        2*a2*b1*b2**2 - 2*a2*b1*b2 - 4*a2*b2**2 + 4*a2*b2 + b1**2*b2**2 - b1**2*b2 - 2*b1*b2**3 - b1*b2**2 + 3*b1*b2 + 4*b2**
        3 - 2*b2**2 - 2*b2,
        b1**4 - 6*b1**3 + 11*b1**2 - 6*b1,
        b1**3*b2 - 6*b1**2*b2 + 11*b1*b2 - 6*b2,
```

Figure 2: Necessary and sufficient conditions for existence of feasible solutions.

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*

$$\begin{cases} argmin_{y_i \in \{0,1)} & y_1 + 2y_2 + 3y_3 + 3y_4, \\ & y_1 + y_2 + 2y_3 + y_4 = 3. \end{cases} \quad (3.2)$$

Figure 3 details the solution.

```
In [1]: from sympy import *

        y1, y2, y3, y4, z = symbols('y1 y2 y3  y4  z')

In [2]: # Problem equational formulation
        eqs = [y1 + 2*y2 + 3*y3 +3*y4 -z,     y1 + y2 + 2*y3 +y4 -3]
        eqs = eqs + [x*(x-1) for x in [y1, y2, y3, y4]]
        print(eqs)

        [y1 + 2*y2 + 3*y3 + 3*y4 - z, y1 + y2 + 2*y3 + y4 - 3, y1*(y1 - 1), y2*(y2 - 1), y3*(y3 - 1), y4*(y4 - 1)]

In [3]: # Computing a Groebner basis
        result = groebner(eqs,  y1, y2, y3, y4, z,  order='lex')
        list(result)

Out[3]: [2*y1 + 2*y3 - z**2 + 11*z - 32,
         y2 + y3 + z**2 - 10*z + 23,
         y3**2 - y3,
         y3*z - 6*y3 - z + 6,
         2*y4 - z**2 + 9*z - 20,
         z**3 - 15*z**2 + 74*z - 120]

In [4]: # Solving the last eqution of the Groebner basis
        solve list(result)[-1], z

        [4, 5, 6]
```

Figure 3: Solving optimization problems with Groebner bases. Although, the cost function is linear here, the method works for any polynomial function.

3.2 A second general method for solving binary optimizations : An important construction that comes with the cost function *f* is the gradient ideal. This is a valuable additional information that we will use in the resolution of the problem ($\mathcal{P}$). Now, because the arguments of the cost function *f* are binary, we need to make sense of the derivation of the function *f*. This is taken care of by the introduction of the function

$$\tilde{f} := f + \sum_{i=0}^{m-1} \alpha_i^2 y_i (y_i - 1), \quad (3.3)$$

where α_i are real numbers with $|\alpha_i| > 1$. We can now go ahead and define *the gradient ideal* of *f* as

$$\tilde{I} := \left\langle \partial_{y_0} \tilde{f}, ..., \partial_{\alpha_{m-1}^2} \tilde{f} \right\rangle. \quad (3.4)$$

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

$$A := \mathbb{Q}\left[y_0, ..., y_{m-1}, \alpha_0, ..., \alpha_{m-1} \right] / \tilde{I} \quad (3.5)$$

Let us define the linear map

$$\begin{aligned} m_{\tilde{f}} : \quad & A \to A \\ & g \to \bar{f}_g \end{aligned} \quad (3.6)$$

Because the number of local minima is finite, the residue algebra A is finite dimensional. Because of this, the following is true[1] :

- The values of $\tilde{f}$, on the set of critical points $\mathcal{V}(\tilde{I})$, are given by the eigenvalues of the matrix $m_{\tilde{f}}$.
- The eigenvalues of m_{y_i} and m_{α_i} give the coordinates of the points of $\mathcal{V}(\tilde{I})$.
- If v is an eigenvector for $m_{\tilde{f}}$, then it is also an eigenvector for m_{y_i} and m_{α_i} for $0 \le i \le m - 1$.

We need to compute a basis for A. This is done by first computing a Groebner basis for $\tilde{I}$ and then extracting the standard monomials (i.e., the monomials in $\mathbb{Q}\left[y_0, ..., y_{m-1}, \alpha_0, ..., \alpha_{m-1} \right]$ 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 $\tilde{I}$ is a Groebner basis with respect to plex(α , *y*).

Example 5 *We illustrate this on*

$f = 2 + 7x_4 + 2x_3 + 2x_4x_3 - 2x_3x_2 - x_1 - 4x_4x_1 - 2x_3x_1 + x_2x_1$

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

$\{1, x_4, x_3, x_4x_3, x_2, x_4x_2, x_3x_2, x_3x_2x_4, x_1, x_4x_1, x_3x_1, x_1x_3x_4, x_2x_1, x_4x_1x_2, x_1x_3x_2, x_1x_3x_2x_4\}$.

The matrix $m_{\tilde{f}}$ is

$$m_{\tilde{f}} = \begin{Bmatrix} 2 & 7 & 2 & 2 & 0 & 0 & -2 & 0 & -1 & -4 & -2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 9 & 0 & 4 & 0 & 0 & 0 & -2 & 0 & -5 & 0 & -2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 9 & 0 & 0 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 13 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & -7 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 7 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & -4 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 & 2 & 1 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 2 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & -2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{Bmatrix}$$

We obtain the following eigenvalues for $m_{\tilde{f}}$:

{ 0, 1, 2, 4, 5, 6, 9, 11, 13}.

This is also the set of values that *f* takes on $\mathcal{V}(\tilde{I})$. 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 $\hat{x} \in \mathcal{V}(\tilde{I})$ that minimizes *f*. The coordinates of the global minimum $\hat{x} = (\hat{x}_0, ..., \hat{x}_{m-1})$ are defined by $m_{x_i} v = \hat{x}_i v$, and this gives $x_1 = x_2 = x_3 = 1$, $x_4 = 0$, and $\alpha_1 = 2\alpha_2 = \alpha_3 = 2, \alpha_4 = 5$.

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[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[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[13]. In fact, in addition to reduction, we can also adjust the coffiecients to be within the dynamic range needed, at the same time. Consider the quadratic polynomial

$$H_{ij} := Q_i P_j + S_{i,j} + Z_{i,j} - S_{i+1,j-1} - 2Z_{i,j+1},$$

with the binary variables P_j, Q_i, $S_{i,j}$, $S_{i+1,j-1}$, $Z_{i,j}$, $Z_{i,j+1}$. 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

$$S = \{H_{ij}\} \cup \{x^2 - x, x \in \{P_j, Q_i, S_{i,j}, S_{i+1,j-1}, Z_{i,j}, Z_{i,j+1}\}\},$$

and look for a positive quadratic polynomial $H_{ij}^+ = \sum_{t \in \beta | deg(t) \le 2} a_t t$ 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

$$t_1 := Q_i P_j + S_{i,j} + Z_{i,j} - S_{i+1,j-1} - 2Z_{i,j+1},$$

$$t_2 := (-Z_{i,j+1} + Z_{i,j})\, S_{i+1,j-1} + (Z_{i,j+1} - 1) Z_{i,j},$$

$$t_3 := (-Z_{i,j+1} + Z_{i,j})\, S_{i,j} + Z_{i,j+1} - Z_{i,j+1} Z_{i,j},$$

$$t_4 := (S_{i+1,j-1} + Z_{i,j+1} - 1)\, S_{i,j} - S_{i+1,j-1} Z_{i,j+1},$$

$$t_5 := (-S_{i+1,j-1} - 2Z_{i,j+1} + Z_{i,j} + S_{i,j}) Q_i - S_{i,j} - Z_{i,j} + S_{i+1,j-1} + 2Z_{i,j+1},$$

$$t_6 := (-S_{i+1,j-1} - 2Z_{i,j+1} + Z_{i,j} + S_{i,j})\, P_j - S_{i,j} - Z_{i,j} + S_{i+1,j-1} + 2Z_{i,j+1},$$

in addition to 3 more cubic polynomials.

We take $H_{ij}^+ = \sum_{t \in \beta | deg(t) \le 2} a_t t$; and solve for the a_t. We can require that the coefficients a_t 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_i, $S_{i,j}$, $S_{i+1,j-1}$, $Z_{i,j}$, and $Z_{i,j+1}$, be within $[1-\varepsilon, 1+\varepsilon]$). The ensemble of these constraints translates

into a simple real optimization problem for the coefficients a_t.

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 ($\mathcal{P}$) 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:

$$(\mathcal{P}_*) : \arg\min_{(y_0,\ldots,y_{m-1})\in\{0,1\}^m} \quad y_0 \sum_{i=1}^{8} c_i y_i . \qquad (5.1)$$

Before we start annealing, we need to map the logical variables y_i to the physical qubits of the hardware. Similarly, the quadratic term $c_i y_0 y_i$ needs to be mapped into a coupling between physical qubits with strength given by the coefficient c_i. 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, ..., 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.

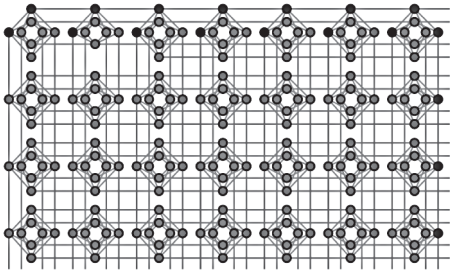

Figure 5: The depicted embedding (for the problem $(\mathcal{P})_*$) has two long chains that don't persist through the adiabatic evolution (in D-Wave 2000Q processor). In this case, the quantum processor fails to return an answer.

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

$$\pi(x_i) = \sum_{ij} \alpha_{ij} y_j \qquad (5.2)$$

with $\sum_{ij} \alpha_{ij} = \beta_i, \alpha_{ij_1}\alpha_{ij_2=0}, \ \alpha_{ij}(\alpha_{ij}-1) = 0$,

where the binary number β_i is 1 if the physical qubits x_i is used and 0 otherwise. We write *domain*(π) = **Vertices**(X) and *support*(π) = **Vertices**(X^{β}) with $X^{\beta} \subset_{subgraph} X$. The fiber of the map π at $y_j \in$ **Vertices**(Y) is given by

$$\pi^{-1}(\gamma_j) = \phi(y_j) = \left\{ x_i \in \textbf{Vertices}(X) \mid \ \alpha_{ij} = 1 \right\} \quad (5.3)$$

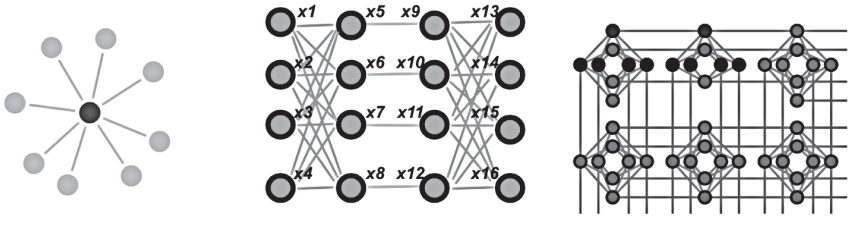

Figure 4: (Left) The logical graph of the objective function in ($\mathcal{P}_*$), can not be embedded inside Chimera graph. (Center) We blow up the central node into edges (x_5, x_9) and redistribute the surrounding nodes. (Right) Embedding inside an actual D-Wave 2000Q quantum annealer; in red, the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

The conditions on the parameters α_{ij} guarantee that fibers don't intersect (i.e., π 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

not go into the details of these conditions, which can be found in[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*

$$\alpha_{1,1}\alpha_{1,2}, \alpha_{1,1}\alpha_{1,3}, \alpha_{1,2}\alpha_{1,3}, \tag{5.4}$$

$$\alpha_{2,1}\alpha_{2,2}, \alpha_{2,1}\alpha_{2,3}, \alpha_{2,2}\alpha_{2,3}, \tag{5.5}$$

$$\alpha_{3,1}\alpha_{3,2}, \alpha_{3,1}\alpha_{3,3}, \alpha_{3,2}\alpha_{3,3}, \tag{5.6}$$

$$\alpha_{4,1}\alpha_{4,2}, \alpha_{4,1}\alpha_{4,3}, \alpha_{4,2}\alpha_{4,3}, \tag{5.7}$$

$$\alpha_{5,1}\alpha_{5,2}, \alpha_{5,1}\alpha_{5,3}, \alpha_{5,2}\alpha_{5,3}, \tag{5.8}$$

and

$$\alpha_{1,1}+\alpha_{1,2}+\alpha_{1,3}-\beta_1, \quad \alpha_{2,1}+\alpha_{2,2}+\alpha_{2,3}-\beta_2,$$

$$\alpha_{3,1}+\alpha_{3,2}+\alpha_{3,3}-\beta_3, \quad \alpha_{4,1}+\alpha_{4,2}+\alpha_{4,3}-\beta_4,$$

$$\alpha_{5,1}+\alpha_{5,2}+\alpha_{5,3}-\beta_5.$$

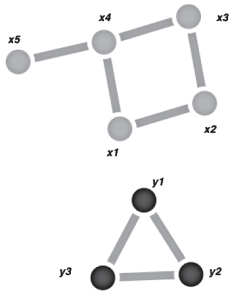

Figure 6: The set of all fiber bundles $\pi : X \to Y$ defines an algebraic variety. This variety is given by the Groebner basis (5.9).

The Pullback Condition reads

$$-1+\alpha_{4,1}\alpha_{5,2}+\alpha_{3,1}\alpha_{4,2}+\alpha_{1,1}\alpha_{2,2}+\alpha_{3,2}\alpha_{4,1}+\alpha_{1,2}\alpha_{2,1}$$
$$+\alpha_{1,2}\alpha_{4,1}+\alpha_{2,2}\alpha_{3,1}+\alpha_{1,1}\alpha_{4,2}+\alpha_{2,1}\alpha_{3,2}+\alpha_{4,2}\alpha_{5,1},$$

$$-1+\alpha_{3,3}\alpha_{4,1}+\alpha_{1,3}\alpha_{2,1}+\alpha_{2,3}\alpha_{3,1}+\alpha_{4,1}\alpha_{5,3}+\alpha_{1,3}\alpha_{4,1}$$
$$+\alpha_{1,1}\alpha_{2,3}+\alpha_{4,3}\alpha_{5,1}+\alpha_{2,1}\alpha_{3,3}+\alpha_{3,1}\alpha_{4,3}+\alpha_{1,1}\alpha_{4,3},$$

$$-1+\alpha_{3,3}\alpha_{4,2}+\alpha_{1,2}\alpha_{2,3}+\alpha_{1,2}\alpha_{4,3}+\alpha_{1,3}\alpha_{2,2}+\alpha_{1,3}\alpha_{4,2}$$
$$+\alpha_{2,3}\alpha_{3,2}+\alpha_{2,2}\alpha_{3,3}+\alpha_{4,2}\alpha_{5,3}+\alpha_{3,2}\alpha_{4,3}+\alpha_{4,3}\alpha_{5,2},$$

Finally, the Connected Fiber Condition is given by

$$-\alpha_{1,1}\alpha_{2,1}\alpha_{5,1}, -\alpha_{1,1}\alpha_{3,1}\alpha_{5,1}, -\alpha_{1,2}\alpha_{2,2}\alpha_{5,2},$$
$$-\alpha_{1,2}\alpha_{3,2}\alpha_{5,2}, -\alpha_{1,3}\alpha_{2,3}\alpha_{5,3}, -\alpha_{1,3}\alpha_{3,3}\alpha_{5,3}$$
$$-\alpha_{2,1}\alpha_{3,1}\alpha_{5,1}, -\alpha_{2,1}\alpha_{4,1}\alpha_{5,1}, -\alpha_{2,2}\alpha_{3,2}\alpha_{5,2},$$
$$-\alpha_{2,2}\alpha_{4,2}\alpha_{5,2}, -\alpha_{2,3}\alpha_{3,3}\alpha_{5,3}, -\alpha_{2,3}\alpha_{4,3}\alpha_{5,3},$$

$$\alpha_{2,1}\alpha_{5,1}\alpha_{2,2}\alpha_{5,2}\alpha_{2,3}\alpha_{5,3}.$$

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

$$\mathcal{B} = \left\{\beta_1 - 1, \beta_2 - 1, \beta_3 - 1, \beta_4 - 1, \beta_5^2 - \beta_5, \alpha_{ij}^2 - \alpha_{ij}, \right. \tag{5.9}$$

$$\alpha_{1,2}\alpha_{1,3}, \alpha_{1,2}\alpha_{3,2}, \alpha_{1,3}\alpha_{3,3}, \alpha_{2,2}\alpha_{2,3}, \alpha_{2,2}\alpha_{4,2},$$
$$\alpha_{2,2}\alpha_{5,2}, \alpha_{2,3}\alpha_{4,3}, \alpha_{2,3}\alpha_{5,3}, \alpha_{3,2}\alpha_{3,3}, \alpha_{4,2}\alpha_{4,3},$$

$$\alpha_{4,2}\alpha_{5,3}, \alpha_{4,3}\alpha_{5,2}, \alpha_{5,2}\alpha_{5,3}, \alpha_{4,2}\alpha_{5,2}$$
$$-\alpha_{5,2}\alpha_{4,2}\beta_5 - \alpha_{5,2}\alpha_{4,3}\alpha_{1,2}\alpha_{5,3} - \alpha_{5,3},$$

$$-\alpha_{2,2}\alpha_{5,3} - \alpha_{3,2}\alpha_{5,3} + \alpha_{1,2}\beta_5 + \alpha_{2,2}\beta_5$$
$$\left. +\alpha_{3,2}\beta_5 + \alpha_{3,3}\beta_5 + \alpha_{5,2} + \alpha_{5,3} - \beta_5 \right\}.$$

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

5.2 Symmetry Reduction : Many of the embeddings aquired 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* :

$$Q_X(x) = x_1x_2 + x_2x_3 + x_3x_4 + x_1x_4 + x_4x_5 \tag{5.10}$$

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 simplied form:

$$Q_X(x, K) = Kx_2 + Kx_4 + x_4x_5 \tag{5.11}$$

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

$$K = \alpha_{01} y_1 + \alpha_{02} y_2 + \alpha_{03} y_3. \quad (5.12)$$

$$x_1 = \alpha_{i1} y_1 + \alpha_{i2} y_2 + \alpha_{i3} y_3 \text{ for } i = 2, 4, 5. \quad (5.13)$$

The table below compares the computations of the surjections π with and without the use of invariants:

	original coords	*invt coords*
Time for computing a Groebner basis (in secs)	0.122	0.039
Number of defining equations	58	30
Maximum degree in the defining eqns	3	2
Number of variables in the defining eqns	20	12
Number of solutions	48	24

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[7], that is used in[8]). As a matter of fact, the procedure which we are about to describe can be used to construct the full *Groebner fan*[5,1] of a given toric ideal. We leave the technical details for a future work. A related notion is the so-called Graver basis which extends toric Groebner bases in the context of convex optimization. A hybrid classical-quantum algorithm for computing Graver bases is given in[17].

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 $a_i = (a_{1i}, ..., a_{ni})^T$ is identified with a Laurent monomial $y^{a_i} = y_1^{a_{1i}} ... y_m^{a_{ni}}$. In this case, the toric ideal $\mathcal{J}_A$ associated with the configuration A is the kernel of the algebra homomorphism

$$\mathbb{Q}[x] \to \mathbb{Q}[y] \quad (6.1)$$

$$x_i \to y^{a_i}. \quad (6.2)$$

From this it follows that the toric ideal $\mathcal{J}_A$ is generated by the binomials $x^{u_+} - x^{u_-}$; where the vector $u = u_+ - u_- \in \mathbb{Z}^{+n} \oplus \mathbb{Z}^{+n}$ runs over all integer vectors in Ker_ZA, 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 $\mathcal{J}_A$.

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 $\mathcal{J}_A$. The example we choose is taken from[1]-Chapter 8. The matrix A is given by

$$A = \begin{pmatrix} 4 & 5 & 1 & 0 \\ 2 & 3 & 0 & 1 \end{pmatrix} \quad (6.3)$$

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 $u = (a, b, -5b - 4a, -3b, -2a)$, which is a linear combination (over $\mathbb{Z}$) of the two vectors. As in[1], we consider the lexographical ordering $plex(w_4, w_3, w_2, w_1)$ represented by the matrix order

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad (6.4)$$

The cost function is given by the square of the Euclidean norm of the vector Mu^t. 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 [$a_{0,+}$, $a_{0,-}$, ..., $b_{2,+}$, $b_{2,-}$], corresponding to the binary decomposition of the integers $a = \sum_{i=0,1,2} \left(a_{i,+} - a_{i,-}\right) 2^i$ and $b = \sum_{i=0,1,2} \left(b_{i,+} - b_{i,-}\right) 2^i$. 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

```
In [54]: HA = get_hardware_adjacency(solver)

         embeddings = find_embedding(Q, HA, verbose=0)
         embeddings

Out[54]: [[315, 317, 443, 571],
          [457, 572, 580, 585, 588],
          [445, 448, 453, 461],
          [313, 318, 441, 569],
          [312, 440, 568],
          [575, 576, 579, 583, 591],
          [447, 455, 456, 463, 584],
          [450, 573, 578, 581, 589],
          [459, 574, 582, 587, 590],
          [444, 451, 452, 460],
          [314, 316, 442, 570],
          [446, 449, 454, 462, 577]]

In [55]: (h0, j0, jc, new_emb) = embed_problem(hQ, jQ, embeddings, HA)
         emb_j = j0.copy()
         emb_j.update(jc)

         solver = conn.get_solver("C16")

         result = solve_qubo(solver,  emb_j, num_reads=10000)
         new_answer = unembed_answer(result['solutions'], new_emb, 'minimize_energy', hQ, jQ)
         print list(set(map(str,new_answer)))
         for key in result.keys()[0:-1]:
             print ("--> "+ key + " : ")
             print("-----------")
             print result[key]

['[1, -1, -1, -1, 1, 1, 1, -1, 1, -1, -1, 1]', '[-1, 1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1]', '[-1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, 1, 1]', '[-1, 1, -1, -1, -1, -1, -1, -1, 1, -1, 1, 1, -1]', '[-1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -
1, -1]', '[-1, 1, -1, -1, 1, -1, -1, -1, -1, 1, -1, -1]', '[-1, -1, -1, 0, 1, -1, 1, -1, -1, -1, -1, 1]', '[1, -1, -
1, -1, 1, -1, -1, -1, -1, -1, 1]', '[-1, -1, -1, -1, 1, -1, 1, -1, -1, -1, -1, 1]']
--> timing :
-----------
{'total_real_time': 1649538, 'qpu_delay_time_per_sample': 21, 'anneal_time_per_run': 20, 'post_processing_overhead_ti
me': 15718, 'qpu_sampling_time': 1640000, 'readout_time_per_run': 123, 'qpu_access_overhead_time': 9721, 'qpu_anneal_
time_per_sample': 20, 'total_post_processing_time': 127210, 'qpu_programming_time': 9538, 'run_time_chip': 1640000,
'qpu_access_time': 1649538, 'qpu_readout_time_per_sample': 123}
--> energies :
-----------
[-9237.0, -9237.0, -9236.0, -9236.0, -9201.333333333332, -9200.333333333332, -9194.0, -9194.0, -9193.0, -9193.0, -916
9.0, -9169.0, -9169.0, -9169.0, -9168.0, -9168.0, -9168.0, -9168.0, -9167.0, -9166.0, -9166.0, -9166.0, -9165.0, -916
5.0, -9159.0, -9159.0, -9158.333333333332, -9158.0, -9158.0, -9158.0, -9157.333333333332, -9156.0, -9156.0, -9155.0, -9155.0,
-9149.0, -9148.0, -9140.0, -9140.0, -9139.0, -9139.0, -9133.333333333332, -9133.333333333332, -9132.333333333332, -91
32.333333333332, -9132.0, -9132.0, -9131.0, -9131.0, -9130.333333333334, -9129.333333333334, -9126.0, -9126.0, -9126.
0, -9126.0, -9125.0, -9125.0, -9125.0, -9125.0, -9125.0, -9124.0, -9123.333333333332, -9123.0, -9123.0, -9123.0, -9122.3333333
33332, -9122.0, -9122.0, -9120.333333333332, -9119.333333333332, -9116.0, -9116.0, -9115.0, -9115.0, -9114.0, -9113.
0, -9113.0, -9113.0, -9112.0, -9112.0, -9106.0, -9105.0, -9104.333333333332, -9103.333333333332, -9101.0, -9101.0, -9
```

Figure 7: Computation of toric Groebner bases on the D-Wave 2000Q quantum processor.

$$\mathcal{B} = \begin{cases} w_2^2 w_3^2 - w_1^3, \\ w_1^4 w_4 - w_2^3 w_3, \\ w_1 w_4 w_3 - w_2, \\ w_2 w_4 w_3^3 - w_1^2, \\ w_3^4 w_4^2 - w_1. \end{cases} \tag{6.5}$$

7. Groebner Bases in the Fundamental Theory of AQC

The role of the so-called *anti-crossings*[4, 19] 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*[20, 21]).

Consider the time dependant Hamiltonian:

$$H(s) = (1 - s)H_{initial} + sH_{final}, \tag{7.1}$$

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

$$f(s, \lambda) = \det\left(H(s) - \lambda I\right). \tag{7.2}$$

where I is the identity $n \times n -$ matrix. The important role that the function f plays in AQC is described in[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[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 λ, 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,

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 $\mathcal{I}$ generated by the two polynomials $\partial_s f$ and $\partial_\lambda f$. It is clear that the variety of $\mathcal{I}$ gives the set of all critical points of f. To capture the non degeneracy, we need to saturate the ideal $\mathcal{I}$ with the polynomial det(Hessian(f)). This saturation is the ideal given by all polynomials in $\mathcal{I}$ that vanish for all the zeros of $\mathcal{I}$ 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 $\text{NormalForm}_{\mathcal{B}}(\det(\text{Hessian}(f)))$ is not zero at p, where $\mathcal{B}$ is a Groebner basis for the ideal $\mathcal{I}$.

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 Castelnuov, 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[24]. 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[25]. Building on various adiabatic theorems of the early quantum mechanics and complexity theory, adiabatic quantum computing was proposed by Farhi *et al* in 2001[2].

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

1. David A. Cox, John B. Little, and Donal O'Shea, *Using algebraic geometry*, Graduate texts in mathematics, Springer, New York, (1998).
2. Edward Farhi, Jerey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda, *A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem*, Science **292**, no. 5516, 472–475 (2001).
3. Wim van Dam, Michele Mosca and Umesh Vazirani, *How powerful is adiabatic quantum computation?* ArXiv:02060C3 (2002).
4. Sei Suzuki, Jun ichi Inoue, and Bikas K. Chakrabarti, *Quantum ising phases and transitions in transverse ising models*, Springer Berlin Heidelberg, (2013).
5. Bernd Sturmfels, *Gröbner bases and convex polytopes*, University Lecture Series, vol. 8, American Mathematical Society, Providence, RI, (1996). MR1363949
6. Pablo A. Parrilo and Bernd Sturmfels, *Minimizing polynomial functions*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science (2001).
7. Pasqualina Conti and Carlo Traverso, *Buchberger algorithm and integer programming*, Proceedings of the 9th International Symposium, on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (London, UK, UK), AAECC-9, Springer-Verlag, (1991), pp. 130–139.
8. Sridhar R. Tayur, Rekha R. Thomas, and N. R. Natraj, *An algebraic geometry algorithm for scheduling in presence of setups and correlated demands*, Math. Program. **69**, 369–401 (1995).

9. Bernd Sturmfels and Rekha R. Thomas, *Variation of cost functions in integer programming*, Math. Program. **77**, 357–387 (1997).
10. Dimitris Bertsimas, Georgia Perakis, and Sridhar Tayur, *A new algebraic geometry algorithm for integer programming*, Management Science **46**, no. 7, 999–1008 (2000).
11. Jean Charles Faugère, *A new efficient algorithm for computing Grnobner bases without reduction to zero (f5)*, Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation (New York, NY, USA), ISSAC '02, ACM, (2002), pp. 75–83.
12. Jean-Charles Faugere, *A new efficient algorithm for computing Gröbner bases (F4)*, Journal of Pure and Applied Algebra **139**, no. 13, 61–88 (1999).
13. Raouf Dridi and Hedayat Alghassi, *Prime factorization using quantum annealing and computational algebraic geometry*, Sci. Rep. 7 (2017).
14. Raouf Dridi, Hedayat Alghassi, and Sridhar Tayur, *A novel algebraic geometry compiling framework for adiabatic quantum computations*, arXiv:1810.01440, (2018).
15. Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy, *Nextgeneration topology of d-wave quantum processors*, D-Wave. (2019).
16. Peter J. Olver, *Classical invariant theor*y, London Mathematical Society Student Texts, (Cambridge University Press), (1999).
17. Hedayat Alghassi, Raouf Dridi, and Sridhar Tayur, *Graver bases via quantum annealing with application to non-linear integer programs*, ArXiv:1902.04215. (2019).
18. Edward Farhi, Jerey Goldstone, and Sam Gutmann, *A quantum approximate optimization algorithm*, ArXiv:1411.4028. (2014).
19. J. von Neumann and E. P. Wigner, *Über das verhalten von eigenwerten bei adiabatischen prozessen*, pp. 294–297, Springer Berlin Heidelberg, Berlin, Heidelberg, (1993).
20. Raoul Bott, *Morse theory indomitable*, Publications Math–ematiques de l'IHÉS **68**, 99–114 (1988) (en). MR90f:58027
21. Edward Witten, *Supersymmetry and Morse theory*, J. Dierential Geom. **17**, no. 4, 661–692 (1982).
22. Raouf Dridi, Hedayat Alghassi and Sridhar Tayur, *Homological description of the quantum adiabatic evolution with a view toward quantum computations,* ArXiv:1811.00675. (2018).
23. Raouf Dridi, Hedayat Alghassi and Sridhar Tayur, *Enhancing the efficiency of adiabatic quantum computations,* ArXiv:1903.01486. (2019).
24. Richard P. Feynman, *Simulating physics with computers*, International Journal of Theoretical Physics **21**, no. 6, 467–488 (1982).
25. P. Ray, B. K. Chakrabarti and Arunava Chakrabarti, *Shefrington-Kirlapatrick model in a transverse field: Absence of replica symmetry breaking due to quantum fluctuations*, Phys. Rev. **B 39**, 11828–11832 (1989).