Quantum Information and Computation, Vol. 23, No. 9&10 (2023) 0721–0732

COMPUTING THE QUANTUM GUESSWORK A QUADRATIC ASSIGNMENT PROBLEM

MICHELE DALL'ARNO
Department of Computer Science and Engineering, Toyohashi University of Technology, Japan
1-1 Hibarigaoka, Tempaku-cho, Toyohashi, Aichi, 441-8580, Japan
Yukawa Institute for Theoretical Physics, Kyoto University
Sakyo-ku, Kyoto, 606-8502, Japan

FRANCESCO BUSCEMI
Department of Mathematical Informatics, Graduate School of Informatics, Nagoya University
Chikusa-Ku, Nagoya, Aichi, 464-8601, Japan

TAKESHI KOSHIBA
Faculty of Education and Integrated Arts and Sciences, Waseda University
Shinjuku-ku, Tokyo, 169-8050, Japan

Received December 21, 2022
Revised June 27, 2023

The quantum guesswork quantifies the minimum number of queries needed to guess the state of a quantum ensemble if one is allowed to query only one state at a time. Previous approaches to the computation of the guesswork were based on standard semi-definite programming techniques and therefore lead to approximated results. In contrast, we show that computing the quantum guesswork of qubit ensembles with uniform probability distribution corresponds to solving a quadratic assignment problem and we provide an algorithm that, upon the input of any qubit ensemble over a discrete ring, after finitely many steps outputs the exact closed-form expression of its guesswork. While in general the complexity of our guesswork-computing algorithm is factorial in the number of states, our main result consists of showing a more-than-quadratic speedup for symmetric ensembles, a scenario corresponding to the three-dimensional analog of the maximization version of the turbine-balancing problem. To find such symmetries, we provide an algorithm that, upon the input of any point set over a discrete ring, after finitely many steps outputs its exact symmetries. The complexity of our symmetries-finding algorithm is polynomial in the number of points. As examples, we compute the guesswork of regular and quasi-regular sets of qubit states.

Keywords: quantum ensemble, quantum measurement, quantum state discrimination, quantum hypothesis testing, quantum guesswork

1 Introduction

We consider the following communication scenario, described in terms of standard concepts and results in quantum information theory [1]. Let an ensemble of quantum states be given. At each round, a referee prepares a state from the ensemble. The task is to guess which state it is, being allowed to query one state at a time until the referee's answer is on the affirmative, at which point a new round begins. The cost function is represented by the average number

of queries needed to correctly guess the state of the ensemble, and is therefore referred to as the *quantum guesswork* [2–15]. Notice that, if multiple states could be queried at a time, the corresponding cost function would instead be the entropy [14] of the ensemble.

The most general strategy consists of a sequence of nondemolishing quantum measurements (quantum instruments), each producing as a classical outcome the next query for the referee. However, as further detailed in Ref. [14], by considering the composition of such instruments, such a strategy reduces to performing a single quantum measurement (on the single copy of the state provided by the referee), whose classical outcomes are represented by tuples of ordered queries for the referee. In other words, one will first query the referee for the state corresponding to the first entry in the output tuple. If the answer is on the negative, one will proceed querying for the second entry in the output tuple, and so on, with the goal of correctly guessing with the minimum number of queries.

The usual approach [13] to compute the guesswork is based on a factorial-size semidefinite program, outputting a numerically approximated result within any given tolerance in polynomial time in the (factorially growing) problem size. Here, instead, we are interested in an exact algorithm to compute the guesswork in finite time, where our computational model is a machine capable of storing integer numbers and of performing additions and multiplication in finite time. Even the existence of such an algorithm is not guaranteed a priori, given that the guesswork problem is, by definition, a continuous optimization problem. For the qubit case, however, the equivalence of the guesswork with a finite optimization problem has recently been shown [15]. Our analysis begins with the observation that such a finite optimization problem is an instance of the (generally NP-hard [16]) quadratic assignment problem [17] (see also [18, 19] for reviews).

Our main result consists in showing how the symmetries of the ensemble, for whose characterization we provide an exact polynomial-time algorithm, can be exploited to achieve a more-than-quadratic speedup in the computation of the guesswork. This scenario corresponds to the three-dimensional analog of the maximization version of the turbine-balancing problem [20] for a particular vector of coefficients, in which blades are ideally symmetrically distributed on a sphere instead of a circle. To illustrate our results, we provide implementations of our symmetries-finding and guesswork-computing algorithms in the C programming language, and we use them to exactly compute the guesswork of regular and quasi-regular ensembles of up to twenty-four states, geometrically corresponding to Platonic and Archimedean solids in the Bloch sphere.

2 Formalization

Qubit states are in one-to-one correspondence with Pauli vectors, that is, three-dimensional vectors within the unit sphere. Hence, an ensemble of N qubit states with uniform probability distribution and without repetitions can be represented by a finite set $\mathcal{V}$ of N Pauli vectors, that is

$$\mathcal{V} \subseteq \left\{ v \in \mathbb{R}^3 \;\middle|\; |v|_2 \leq 1 \right\}.$$

Following Ref. [21], we denote with $\mathcal{V}^{\underline{N}}$ the set of N-tuples on $\mathcal{V}$ without repetitions, that is $\mathbf{v} \in \mathcal{V}^{\underline{N}}$ if and only if

$$\mathbf{v} = (v_i)_{i\in\{1,\dots,N\}} \text{ s.t. } \bigcup_{i\in\{1,\dots,N\}} \{v_i\} = \mathcal{V}.$$

Any qubit effect can be represented as an affine function from the set of qubit states to $[0,1]$. Hence, any $\mathcal{V}^{\underline{N}}$ valued qubit measurement can be represented as a family $(\pi_{\mathbf{v}})_{\mathbf{v}\in\mathcal{V}^{\underline{N}}}$ of such affine functions indexed by $\mathcal{V}^{\underline{N}}$ such that $\sum_{\mathbf{v}\in\mathcal{V}^{\underline{N}}} \pi_{\mathbf{v}} = u$, where u is the unit effect, that is

$$\mathcal{P} = (\pi_{\mathbf{v}} : \mathcal{S} \to [0,1])_{\mathbf{v}\in\mathcal{V}^{\underline{N}}} \text{ s.t. } \sum_{\mathbf{v}\in\mathcal{V}^{\underline{N}}} \pi_{\mathbf{v}} = u.$$

For any such qubit ensemble $\mathcal{V}$ and any $\mathcal{V}^{\underline{N}}$-valued qubit measurement $\mathcal{P}$, the *guesswork* $G(\mathcal{V},\mathcal{P})$ is the number of queries needed on average to correctly guess the state of $\mathcal{V}$ when measuring $\mathcal{P}$, that is (see Ref. [15] for more details)

$$G(\mathcal{V},\mathcal{P}) := \frac{1}{N} \sum_{\substack{\mathbf{v}\in\mathcal{V}^{\underline{N}} \\ i\in\{1,\dots,N\}}} \pi_{\mathbf{v}}(v_i)\ i,$$

where v_i denotes the i-th component of vector v. The *minimum guesswork* $G_{\min}(\mathcal{V})$ is the minimum of the guesswork $G(\mathcal{V},\mathcal{P})$ over any $\mathcal{V}^{\underline{N}}$-valued measurement $\mathcal{P}$, that is

$$G_{\min}(\mathcal{V}) := \min_{\mathcal{P}} G(\mathcal{V},\mathcal{P}). \tag{1}$$

3 Main result

Our first observation is that the optimization problem in Eq. (1) corresponds to a specific instance of the quadratic assignment problem. Since the former consists of an optimization over a continuous set, the latter represents its closed-form solution.

Lemma 1. *For any ensemble $\mathcal{V}$ of N Pauli vectors with uniform probability distribution, the minimum guesswork $G_{\min}(\mathcal{V})$ is given by the following quadratic assignment problem*

$$G_{\min}(\mathcal{V}) = \frac{1}{2}\left(N + 1 - \frac{1}{N}\sqrt{\max_{X\in\mathrm{Perm}(N)} \mathrm{Tr}\left[WXDX^T\right]}\right),$$

where $\mathrm{Perm}(N)$ *denotes the set of $N\times N$ permutation matrices, D is the $N\times N$ matrix given by $D = dd^T$, in turn d is the N-dimensional column vector whose i-th entry is given by $2i - N + 1$, W is the $N\times N$ Gram matrix given by $W := V^TV$, and in turn V is any $3\times N$ matrix whose columns are given by the elements of $\mathcal{V}$ without repetitions.*

Proof. From Theorem 1 and Corollary 1 of Ref. [15] it immediately follows that

$$G_{\min}(\mathcal{V}) = \frac{1}{2}\left(N + 1 - \frac{1}{N}\sqrt{\max_{\mathbf{v}\in\mathcal{V}^{\underline{N}}} |\langle\mathbf{v}\rangle|_2^2}\right), \tag{2}$$

where for any N-tuple $\mathbf{v}$ of elements of $\mathcal{V}$ without repetitions the function $\langle\mathbf{v}\rangle$ is given by

$$\langle\mathbf{v}\rangle := \sum_{i=1}^{N} (2i - N + 1)\, v_i. \tag{3}$$

The result then follows by observing that, by explicit computation, one has

$$\max_{\mathbf{v}\in\mathcal{V}^{\underline{N}}} |\langle \mathbf{v}\rangle|_2^2 = \max_{X\in \mathrm{Perm}(N)} \mathrm{Tr}\left[WXDX^T\right].$$

□

To begin with, let us analyze the complexity of solving this quadratic assignment problem with a naive exhaustive search. The complexity of the exhaustive search is given by the product of the complexity of generating all tuples $\mathbf{v}$ in $\mathcal{V}^{\underline{N}}$ and the complexity of computing the function $\langle \mathbf{v}\rangle$ for each tuple. According to Eq. (3), the complexity of computing $\langle \mathbf{v}\rangle$ is $O(N)$. This can be improved if the tuples are generated so that each new tuple is obtained from the previous one by swapping only two elements. This is possible, for example by means of Heap's or Johnson-Trotter's algorithm [21,22]. In this case, $\langle \mathbf{v}\rangle$ does not need be entirely recomputed at each step following Eq. (3): its value can be stored, and after each new tuple is generated, $\langle \mathbf{v}\rangle$ can be updated by only taking into account the contributions of the two states affected by the swapping. From Eq. (3), if tuples $\mathbf{v}$ and $\mathbf{v}'$ differ by the swap of elements i and j with $i<j$, then we have

$$\langle \mathbf{v}'\rangle = \langle \mathbf{v}\rangle + 2\,(j-i)\,(v_i - v_j)\,, \tag{4}$$

that represents a constant-time update of $\langle \mathbf{v}\rangle$. Hence, the complexity of the algorithm is $O(N!)$. Since this value also corresponds to the cardinality of set $\mathcal{V}^{\underline{N}}$, it also represents the optimal complexity for an exhaustive search. By denoting with nextJT the function that returns the pair of indexes (i,j) that need to be swapped to generate the next tuple according to Heap's or Johnson-Trotter algorithm [21,22] and false if the end of the algorithm has been reached, we arrive at Algorithm 1.

Algorithm 1 Guesswork (naive exhaustive search)

Require: finite set $\mathcal{V}$ of N Pauli vectors
Ensure: $G_{\min}(\mathcal{V}) = (N+1-\sqrt{g}/N)/2$
  $\mathbf{v} \in \mathcal{V}^{\underline{N}}$
  $\overline{v} \leftarrow \langle \mathbf{v}\rangle$
  $g \leftarrow 0$
  while $(i,j) \leftarrow \mathrm{nextJT}(\mathbf{v})$ **do**
    $v_i \leftrightarrow v_j$
    $\overline{v} \leftarrow \overline{v} + 2(j-i)(v_i - v_j)$
    $g \leftarrow \max\left(g, |\overline{v}|_2^2\right)$
  end while
  return g

Our main result consists in showing that, in the presence of symmetries and specifically if set $\mathcal{V}$ is centrally symmetric or vertex transitive, the complexity of Algorithm 1 can be reduced by a more-than-quadratic factor. This scenario corresponds to the three-dimensional analog of the maximization version of the turbine-balancing problem [20] for a particular vector of coefficients, in which blades are ideally symmetrically distributed on a sphere instead of a circle.

A set $\mathcal{V}$ of Pauli vectors is centrally symmetric if and only if for any $\mathbf{v} \in \mathcal{V}$ also $-\mathbf{v} \in \mathcal{V}$. A set $\mathcal{V}$ is vertex transitive if and only if for any pair $\{\mathbf{v}_0, \mathbf{v}_1\} \subseteq \mathcal{V}$, there exists orthogonal matrix O such that $\mathbf{v}_1 = O\mathbf{v}_0$ and $\mathcal{V} = O\mathcal{V}$. We postpone to Section 5 the discussion of a polynomial-time algorithm that exactly computes the symmetries of $\mathcal{V}$.

For any N-tuple $\mathbf{v} = (v_1, \ldots, v_N)$, let us define the reversed N-tuple $\overline{\mathbf{v}} := (v_N, \ldots, v_1)$ and the opposite N-tuple $-\mathbf{v} := (-v_1, \ldots, -v_N)$, and let $\mathcal{T} \subseteq \mathcal{V}^{\underline{N}}$ denote the set of tuples in which each pair of centrally symmetric vectors appear in symmetric positions within the tuple, that is

$$\mathcal{T} := \left\{ \mathbf{v} \in \mathcal{V}^{\underline{N}} \;\middle|\; -\mathbf{v} = \overline{\mathbf{v}} \right\}.$$

Moreover, for any $v \in \mathcal{V}$ let $\mathcal{T}_v \subseteq \mathcal{V}^{\underline{N}}$ denote the set of tuples with fixed point $v_N = v$, that is

$$\mathcal{T}_v := \left\{ \mathbf{v} \in \mathcal{V}^{\underline{N}} \;\middle|\; v_N = v \right\}.$$

The following result holds.

Lemma 2 (Symmetries)**.** *For any given set $\mathcal{V}$ of N Pauli vectors, if $\mathcal{V}$ is centrally symmetric or vertex transitive, there exists a tuple $\mathbf{v}$ attaining the maximum in Eq.* (2) *such that $\mathbf{v} \in \mathcal{T}$ or $\mathbf{v} \in \mathcal{T}_v$, respectively. Moreover, if $\mathcal{V}$ is centrally symmetric and vertex transitive, there exists a tuple $\mathbf{v} \in \mathcal{T} \cap \mathcal{T}_v$ attaining the maximum in Eq.* (2)*.*

Proof. Let us consider the centrally symmetric case. By specializing Lemma 4 of Ref. [15] to the case of qubit ensemble with uniform probability distribution, one has

$$\max_{\mathbf{v} \in \mathcal{V}^{\underline{N}}} |\langle \mathbf{v} \rangle|_2^2 = \max_{\mathbf{v} \in \mathcal{T}} |\langle \mathbf{v} \rangle|_2^2 ,$$

that proves the statement.

Let us consider the vertex transitive case. By definition of vertex transitivity, one has that the range of the squared norm $|\cdot|_2^2$ is unchanged if the range is restricted from $\mathcal{V}^{\underline{N}}$ to $\mathcal{T}_v$, that is

$$\left|\langle \mathcal{V}^{\underline{N}} \rangle\right|_2^2 = |\langle \mathcal{T}_v \rangle|_2^2 ,$$

that proves the statement.

Let us consider the centrally symmetric and vertex transitive case. Due to central symmetry, there exists tuple $\mathbf{v}$ attaining the maximum in Eq. (2) such that $\mathbf{v} \in \mathcal{T}$. Due to vertex transitivity, every tuple $\mathbf{v}$ is unitarily equivalent to a tuple in $\mathcal{T}_v$. Hence the statement is proved. □

The set $\mathcal{T}$ can be generated as follows. Let set $\mathcal{V}' \subseteq \mathcal{V}$ be any subset of the set $\mathcal{V}$ of states, containing one element for each pair of centrally symmetric elements, that is, for each $v \in \mathcal{V}$ either v or $-v$ is in $\mathcal{V}'$, but not both. Let also set $\overline{\mathcal{V}}'$ be the complement of set $\mathcal{V}$, that is $\overline{\mathcal{V}'} := \mathcal{V} \setminus \mathcal{V}'$ or equivalently $\overline{\mathcal{V}'} = -\mathcal{V}'$. First, we show that sets $\mathcal{T}$ and $\{-1, +1\}^{N/2} \times \mathcal{V}'^{\underline{N/2}}$ are in one-to-one correspondence. For any tuple $\mathbf{v}$ in $\mathcal{T}$, one has that tuples $\boldsymbol{\tau}$ and $\mathbf{v}'$ given by

$$\tau_i := \begin{cases} +1 & \text{if } v_i \in \mathcal{V}', \\ -1 & \text{otherwise,} \end{cases}$$

and

$$v_i := \begin{cases} v_i & \text{if } v_i \in \mathcal{V}', \\ -v_i & \text{otherwise,} \end{cases}$$

for any i in $\{1, \ldots, N/2\}$, are in $\{-1, +1\}^{N/2}$ and $\mathcal{V}'^{\underline{N/2}}$, respectively. Vice-versa, for any tuple $\boldsymbol{\tau} \in \{-1, +1\}^{N/2}$ and any tuple $\mathbf{v}' \in \mathcal{V}'^{\underline{N/2}}$ one has that tuple

$$\mathbf{v} := \boldsymbol{\tau} \odot \mathbf{v}' \oplus \overline{-\boldsymbol{\tau} \odot \mathbf{v}'} \tag{5}$$

is in $\mathcal{T}$, where $\odot$ and $\oplus$ denote the Hadamard (that is, element-wise) product and the direct sum (that is, concatenation) of tuples, respectively. Moreover, due to Eq. (5) any element of set $\mathcal{T}$ can be generated from the corresponding element of set $\{-1, +1\}^{N/2} \times \mathcal{V}'^{\underline{N/2}}$ in constant time. The complexities [21, 22] of generating the elements of set $\{-1, 1\}^{N/2}$ and of set $\mathcal{V}'^{\underline{N/2}}$ are $O(2^{N/2})$ and $O((N/2)!)$, respectively. Moreover, through a Gray code [21] it is possible to iteratively generate all tuples $\boldsymbol{\tau}$ in $\{-1, 1\}^{N/2}$ such that each new tuple $\boldsymbol{\tau}'$ differs from the previous tuple $\boldsymbol{\tau}$ by a single sign flip. If the sign flip occurs in the i-th position, from Eq. (3) it immediately follows that function $\langle \mathbf{v} \rangle$ is updated as follows

$$\langle \mathbf{v}' \rangle_N = \langle \mathbf{v} \rangle_N - 4\,(2i - N + 1)\, v_i, \tag{6}$$

that represents a constant-time update. Since for even N one has $N!! = 2^{N/2}(N/2)!$ (with $N!! := N(N-2)(N-4)\ldots$ we denote the double factorial function), central symmetry can be exploited to reduce the complexity of Algorithm 1 by a factor of $(N-1)!!$.

The set $\mathcal{T}_v$ can be generated as follows. First, notice that sets $\mathcal{T}_v$ and $(\mathcal{V} \setminus v)^{\underline{N-1}}$ are in one-to-one correspondence. For any tuple $\mathbf{v}$ in $\mathcal{T}_v$, one has that its restriction $\mathbf{v}' := \mathbf{v}|_{\{1,\ldots,N-1\}}$ is in $(\mathcal{V} \setminus v)^{\underline{N-1}}$. Vice-versa, for any tuple $\mathbf{v}'$ in $(\mathcal{V} \setminus v)^{\underline{N-1}}$, one has that its extension

$$v_i := \begin{cases} v'_i & \text{if } i \in \{1, \ldots, N-1\}, \\ v & \text{otherwise,} \end{cases} \tag{7}$$

is in $\mathcal{T}_v$. Moreover, due to Eq. (7), any element of set $\mathcal{T}_v$ can be generated from the corresponding element of $(\mathcal{V} \setminus v)^{\underline{N-1}}$ in constant time. Since all the elements of $(\mathcal{V} \setminus v)^{\underline{N-1}}$ can be generated in $(N-1)!$ steps [21, 22], vertex transitivity can be exploited to reduce the complexity of Algorithm 1 by a factor of N.

The set $\mathcal{T} \cap \mathcal{T}_v$ can be generated by concatenating the two previous methods. Assuming $\mathcal{V}$ is centrally symmetric and vertex transitive, for any $v \in \mathcal{V}$, let $\mathcal{V}' \subseteq \mathcal{V} \setminus \{v, -v\}$ be such that for any $v' \in \mathcal{V}$ one has $-v' \notin \mathcal{V}'$. Then, sets $\mathcal{T} \cap \mathcal{T}_v$ and $\{-1, 1\}^{N/2-1} \times \mathcal{V}'^{\underline{N/2-1}}$ are in one-to-one correspondence. The correspondence can be explicitly built as before. Moreover, any element of $\mathcal{T} \cap \mathcal{T}_v$ can be generated from the corresponding element of $\{-1, 1\}^{N/2-1} \times \mathcal{V}'^{\underline{N/2-1}}$ in constant time. Since the complexities [21, 22] of generating the elements of sets $\{-1, 1\}^{N/2-1}$ and $\mathcal{V}'^{\underline{N/2-1}}$ are $O(2^{N/2-1})$ and $O((N/2-1)!)$, respectively, central symmetry and vertex transitivity can be exploited to reduce the complexity of Algorithm 1 by a factor of $N(N-1)!!$, that represents a more-than-quadratic speedup. By denoting with nextGray the function that returns the index k that needs to be flipped to generate the next tuple according to the Gray code [21] and false if one cycle of the code has been completed and the algorithm is back to the first tuple produced, we arrive at Algorithm 2.

Table 1 summarizes the results of this section.

Algorithm 2 Guesswork (more-than-quadratic speedup)

Require: finite centrally symmetric and vertex invariant set $\mathcal{V}$ of N Pauli vectors
Ensure: $G_{\min}(\mathcal{V}) = (N + 1 - \sqrt{g}/N)/2$
  $\mathbf{v} \in \mathcal{V}^{\underline{N/2-1}}$ s.t. $\forall v \in \mathbf{v}$ one has $-v \notin \mathbf{v}$
  $\overline{v} \leftarrow \langle \mathbf{v} \rangle$
  $g \leftarrow 0$
  while $(i, j) \leftarrow \text{nextJT}(\mathbf{v})$ **do**
    $v_i \leftrightarrow v_j$
    $\overline{v} \leftarrow \overline{v} + 2(j - i)(v_i - v_j)$
    $\boldsymbol{\tau} \in \{-1, 1\}^{N/2-1}$
    while $k \leftarrow \text{nextGray}(\boldsymbol{\tau})$ **do**
      $v_k \leftarrow -v_k$
      $\overline{v} \leftarrow \overline{v} - 4(2k - N + 1)v_k$
      $g \leftarrow \max(g, |\overline{v}|_2^2)$
    end while
  end while
  return g

Table 1: Complexity of Algorithms 1 and 2 for the exact computation of the minimum guesswork of any given qubit ensemble $\mathcal{V}$ with uniform probability distribution, as a function of the symmetries of $\mathcal{V}$.

Symmetries	Complexity	Speedup (with respect to no symmetries)
No symmetries	$O(N!)$	1
Central symmetry	$O(N!!)$	$(N-1)!!$
Vertex transitivity	$O((N-1)!)$	N
Central symm. & vertex trans.	$O((N-2)!!)$	$N(N-1)!!$

4 Explicit examples

In this section we apply Algorithms 2 and 4 to compute the exact closed-form expression for the minimum guesswork of regular and quasi-regular qubit ensembles up to twenty-four vertices. We also provide [23] a C language implementation of such algorithms.

First, let us discuss ensembles of Pauli vectors whose coordinates can be represented (up to a scaling) by the ring of integers. These are the tetrahedron, octahedron, cube, truncated tetrahedron, cuboctahedron, and truncated octahedron. The values of the guesswork for tuples of Pauli vectors on the ring of integers are reported in Table 2.

Second, let us discuss ensembles of Pauli vectors whose coordinates can be represented (up to a scaling) by the ring

$$p_k(\mathbf{z}) := z_0 + \sqrt{k} z_1,$$

where $\mathbf{z} = (z_0, z_1) \in \mathbb{Z}^2$ and k is a positive non square integer constant that only depends on the ensemble. These are the icosahedron, dodecahedron, truncated cube, and rhombicuboctaedron. To apply Algorithm 4 on a machine that implements integer arithmetic, we need to derive integer formulae for the arithmetic operations of sum, difference, multiplication by an

Table 2: Exact closed-form expression and approximate numerical value of the minimum guesswork of regular and quasi-regular tuples of qubit states on the ring of integers, as given by Algorithm 2.

$\mathcal{V}$	N	g	$G_{\min}$
Tetrahedron	4	$\frac{80}{3}$	~ 1.8545
Octahedron	6	140	~ 2.5140
Cube	8	$\frac{1344}{3}$	~ 3.1771
Truncated tetrahedron	12	$\frac{25168}{11}$	~ 4.5070
Cuboctahedron	12	$\frac{4560}{2}$	~ 4.5104
Truncated octahedron	24	$\frac{183440}{5}$	~ 8.5096

integer, and square. They are clearly given by

$$p_k(\mathbf{z}) \pm p_k(\mathbf{z}') = p_k(z_0 \pm z_0', z_1 \pm z_1'),$$

$$z\, p_k(\mathbf{z}) = p_k(z\mathbf{z}),$$

and

$$p_k(\mathbf{z})^2 = p_k\left(z_0^2 + k z_1^2, 2 z_0 z_1\right).$$

We also need an integer formula to compare numbers. By direct inspection one has $p_k(\mathbf{z}) \geq 0$ if and only if

$$\left(z_0 \geq 0 \text{ and } z_0^2 \geq k z_1^2\right) \text{ or } \left(z_1 \geq 0 \text{ and } z_0^2 \leq k z_1^2\right).$$

The values of the guesswork for tuples of Pauli vectors on this ring are reported in Table 3.

Table 3: Exact closed-form expression and approximate numerical value of the minimum guesswork of regular and quasi-regular tuples of qubit states on the ring $z_0 + \sqrt{k} z_1$, as given by Algorithm 2.

$\mathcal{V}$	N	g	$G_{\min}$
Icosahedron	12	$\frac{16544+7392\sqrt{5}}{10+2\sqrt{5}}$	~ 4.5081
Dodecahedron	20	$\frac{106272+47456\sqrt{5}}{12}$	~ 7.1741
Truncated cube	24	$\frac{47040+23168\sqrt{2}}{5-2\sqrt{2}}$	~ 8.5062
Rhombicuboctahedron	24	$\frac{146128+100128\sqrt{2}}{5+2\sqrt{2}}$	~ 8.5059

5 An exact symmetries finding algorithm

In this section we present an algorithm that, upon the input of any arbitrary-dimensional complex point set, after finitely many-steps outputs its exact symmetries. The complexity of our algorithm is polynomial in the number of points.

Previous works [24–27] approached the problem of finding the symmetries of any given point set (and the related problem of testing the congruence of two sets) from the geometric

viewpoint, that is, by looking for unitary transformations that act as permutations of the set. As a consequence, previous symmetries-finding algorithms depend on the full field structure (in particular, they depend on the arithmetic operation of division). To this aim, they assume the *real computational model*, that is, an unphysical machine that can exactly store any real number and can exactly perform arithmetic, trigonometric, and other functions over reals in finite time.

We instead approach the symmetries-finding problem from the viewpoint of combinatorics, that is, by looking for permutations of the set that act as unitary transformations. In fact, by using well-known results on Gram matrices, we avoid explicitly dealing with unitary transformations altogether. This way, we present a symmetries-finding algorithm (that can also be trivially adapted to congruence-testing) that only depends on the weaker ring structure (that is, division is not assumed). Ours is therefore an *integer computational model* that solely assumes the ability to store integer numbers and to perform additions and multiplication in finite time, thus allowing us to achieve closed-form analytical results on physical machines.

The factorial growth of the number of permutations in the cardinality N of the set dooms to factorial complexity any algorithm based on a naive exhaustive search. However, by exploiting a well-known rigidity property of simplices, we show that without loss of generality it suffices to search over a polynomial-sized subset of permutations. The complexity of our symmetries-finding algorithm is therefore $O(N^{d+2})$, where d denotes the dimension of the complex space.

For any given arbitrary-dimensional spanning set $\mathcal{V}$ of complex vectors, we denote with $\mathrm{Sym}(\mathcal{V})$ the group of permutations of $\mathcal{V}$. A permutation σ in $\mathrm{Sym}(\mathcal{V})$ is called a *geometric symmetry* (in the following, symmetry for short) of $\mathcal{V}$ if and only there exists unitary transformation $\mathcal{U}$ such that

$$\sigma(\mathbf{v}) = \mathcal{U}(\mathbf{v}), \tag{8}$$

where $\mathbf{v} \in \mathcal{V}^{\underline{N}}$ denotes an N-tuple on $\mathcal{V}$ without repetitions. We denote the group of all symmetries of $\mathcal{V}$ with $\mathrm{Geom}(\mathcal{V})$. Notice that the fact that $\mathcal{V}$ is a spanning set guarantees that the mapping between σ and $\mathcal{U}$ satisfying Eq. (8) is bijective.

Since the computation of the unitary transformation $\mathcal{U}$ in Eq. (8) requires divisions in general (take for example $\mathbf{v}$ to be the vertices of a square and $\mathcal{U}$ to correspond with a $\pi/2$ rotation), we need an approach where $\mathcal{U}$ remains implicit. It is a well-known fact that two tuples of vectors are unitarily related if and only if their Gram matrices coincide, where the Gram matrix $G(\mathbf{v})$ of an N-tuple $\mathbf{v} := (\mathbf{v}_i)_i$ of vectors is the $N \times N$ matrix whose (i,j)-th entry is given by the inner product $\mathbf{v}_i \cdot \mathbf{v}_j$, that is

$$[G(\mathbf{v})]_{i,j} = \mathbf{v}_i \cdot \mathbf{v}_j. \tag{9}$$

Hence, tuples $\sigma(\mathbf{v})$ and $\mathbf{v}$ satisfy Eq. (8) if and only if

$$G(\sigma(\mathbf{v})) = G(\mathbf{v}). \tag{10}$$

Since Eq. (9) can be clearly computed without divisions, this observation immediately leads to a division-free exact symmetries-finding algorithm through an exhaustive search over the set $\mathrm{Sym}(\mathcal{V})$ of permutations. The complexity of the exhaustive search is given by the product of the complexity of generating all permutations $\sigma \in \mathrm{Sym}(\mathcal{V})$ and the complexity

of computing and comparing the corresponding Gram matrices. According to Eq. (9), the latter complexity is $O(N^2)$. Hence, the complexity of the algorithm is $O(N!N^2)$. By denoting with nextJT the function that returns the pair of indexes (i, j) that need to be swapped to generate the next tuple according to Heap's or Johnson-Trotter's algorithm [21] and false if the end of the algorithm has been reached, we arrive at Algorithm 3.

Algorithm 3 Symmetries finding (exhaustive search)

Require: d-dimensional spanning set $\mathcal{V}$ of N complex vectors
Ensure: $\mathcal{S} = \mathrm{Geom}(\mathcal{V})(\mathbf{v})$, for some $\mathbf{v} \in \mathcal{V}^{\underline{N}}$
  $\mathbf{v} \in \mathcal{V}^{\underline{N}}$
  $\mathbf{v}' \leftarrow \mathbf{v}$
  $\mathcal{S} \leftarrow \emptyset$
  while $(i, j) \leftarrow \mathrm{nextJT}(\mathbf{v})$ **do**
    $v'_i \leftrightarrow v'_j$
    if $G(\mathbf{v}') = G(\mathbf{v})$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{v}'$
    end if
  end while
  return $\mathcal{S}$

We proceed now to improve the complexity of Algorithm 3 from factorial to polynomial. Let d-tuple $\mathbf{e} := (e_i)_{i=1}^{d} \in \mathcal{V}^{\underline{d}}$ on $\mathcal{V}$ without repetitions be a basis, that is, the determinant $\det G(\mathbf{e})$ of its Gram matrix is non null. Division-free algorithms for the computation of the determinant are known; for a particularly simple one, see Ref. [28]. Due to Eq. (10), a necessary condition for any permutation σ of $\mathcal{V}$ to be a symmetry is that

$$G\left(\sigma\left(\mathbf{e}\right)\right) = G\left(\mathbf{e}\right). \tag{11}$$

For any d-tuple $\mathbf{e}'$ satisfying $G(\mathbf{e}') = G(\mathbf{e})$, the permutation σ such that Eq. (10) holds, if it exists, is unique and can be explicitly derived as follows.

For some basis $\mathbf{e}$ and any vectors $v_0, v_1 \in \mathbb{C}^d$ we say $v_0 \prec_{\mathbf{e}} v_1$ if and only if

$$e_k \cdot (v_0 - v_1) \leq 0,$$

where k is the minimum over $\{1, \ldots, d\}$ such that $e_k \cdot (v_0 - v_1)$ is not null, and we call $\mathbf{e}$-order the order induced by $\prec_{\mathbf{e}}$. Since $\mathbf{e}$ is a basis, $\mathbf{e}$-order is total. Let $\mathbf{v} \in \mathcal{V}^{\underline{N}}$ be the $\mathbf{e}$-ordered N-tuple on $\mathcal{V}$ without repetitions. Due to Eq. (8), for any k in $\{1, \ldots, d\}$ and any i in $\{1, \ldots, N\}$, the inner product $e_k \cdot v_i$ equals the inner product $\sigma(e_k) \cdot \sigma(v_i)$. Hence, tuple $\sigma(\mathbf{v})$ is the $\sigma(\mathbf{e})$-ordered N-tuple of all the elements of $\mathcal{V}$ without repetitions. This explicitly and uniquely identifies permutation σ.

The complexity of the algorithm is given by the product of the complexity of generating all d-tuples in $\mathcal{V}^{\underline{d}}$ and the complexity of processing each tuple. Since the combinations of d elements out of N are $\binom{N}{d}$, and for each combination there are $d!$ differently ordered tuples, the complexity [21, 22] of generating the d-tuples is $O(\binom{N}{d})$, that is, a polynomial of degree d in N. The complexity of processing each tuple is the sum of the complexity of generating $\mathbf{r}'$-ordered N-tuple $\mathbf{v}'$, computing the Gram matrix $G(\mathbf{v}')$, and comparing it with $G(\mathbf{v})$, hence

it is $O(N^2)$. Hence, the complexity of the algorithm is $O(N^{d+2})$. By denoting with nextChase the function that returns the pair of indexes (i, j) that need to be swapped to generate the next combination according to the Chase's sequence [21] and false if the end of the algorithm has been reached, we arrive at Algorithm 4.

Algorithm 4 Symmetries finding (polynomial time)

Require: d-dimensional spanning set $\mathcal{V}$ of N complex vectors
Ensure: $\mathcal{S} = \mathrm{Geom}(\mathcal{V})(\mathbf{v}')$, for some $\mathbf{v}' \in \mathcal{V}^{\underline{N}}$
  $\mathbf{v} \leftarrow \mathbf{e}$-ordered N-tuple in $\mathcal{V}^{\underline{N}}$
  $\mathcal{S} \leftarrow \emptyset$
  while $(i, j) \leftarrow \mathrm{nextChase}(\mathbf{v}, \mathbf{e}')$ **do**
    $e'_i \leftarrow v_j$
    while $(k, l) \leftarrow \mathrm{nextJT}(\mathbf{e}')$ **do**
      $e'_k \leftrightarrow e'_l$
      if $G(\mathbf{e}') = G(\mathbf{e})$ **then**
        $\mathbf{v}' \leftarrow \mathbf{e}'$-ordered N-tuple in $\mathcal{V}^{\underline{N}}$
        if $G(\mathbf{v}') = G(\mathbf{v})$ **then**
          $\mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{v}'$
        end if
      end if
    end while
  end while
  return $\mathcal{S}$

6 Conclusion

We showed that the computation of the guesswork of qubit ensembles with uniform probability distribution corresponds to a quadratic assignment problem. We presented a division-free algorithm for the exact analytical computation of the guesswork with a more-than-quadratic speedup in the presence of symmetries, that is, for the three-dimensional analog of the maximization version of the turbine-balancing problem. As examples, we computed the exact closed-form expression for the guesswork of regular and quasi-regular ensembles of qubit states.

Acknowledgments

This work is dedicated to the memory of Takeshi Koshiba. M. D. acknowledges support from the Department of Computer Science and Engineering, Toyohashi University of Technology, the MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant No. JPMXS0118067285, JSPS KAKENHI Grant Number JP20K03774, and the International Research Unit of Quantum Information, Kyoto University. F. B. acknowledges support from MEXT Quantum Leap Flagship Program (MEXT QLEAP) Grant No. JPMXS0120319794; from MEXT-JSPS Grant-in-Aid for Transformative Research Areas (A) Extreme Universe, No. 21H05183; from JSPS KAKENHI Grants No. 20K03746 and No. 23K03230.

References

1. M. M. Wilde, *Quantum Information Theory*, (Cambridge University Press, 2017).

2. J. Massey, *Guessing and entropy*, Proceedings of 1994 IEEE International Symposium on Information Theory, 204 (1994).
3. E. Arikan, *An inequality on guessing and its application to sequential decoding*, IEEE Trans. Inform. Theory **42**, 99 (1996).
4. E. Arikan and N. Merhav, *Guessing subject to distortion*, IEEE Trans. Inform. Theory **44**, 1041 (1998).
5. E. Arikan and N. Merhav, *Joint source-channel coding and guessing with application to sequential decoding*, IEEE Trans. Inform. Theory **44**, 1756 (1998).
6. J. Pliam, *The Disparity between Work and Entropy in Cryptology*, Cryptology ePrint Archive 1998/024 (1998).
7. D. Malone and W. Sullivan, *Guesswork and Entropy*, IEEE Trans. Inform. Theory **50**, 525 (2004).
8. R. Sundaresan, *Guessing Under Source Uncertainty*, IEEE Trans. Inform. Theory **53**, 269 (2007).
9. M. K. Hanawal and R. Sundaresan, *Guessing Revisited: A Large Deviations Approach*, IEEE Trans. Inform. Theory **57**, 70 (2011).
10. M. M. Christiansen and K. R. Duffy, *Guesswork, Large Deviations, and Shannon Entropy*, IEEE Trans. Inform. Theory **59**, 796 (2013).
11. I. Sason and S. Verdú, *Improved Bounds on Lossless Source Coding and Guessing Moments via Rnyi Measures*, IEEE Trans. Inform. Theory **64**, 4323 (2018).
12. I. Sason, *Tight Bounds on the Rnyi Entropy via Majorization with Applications to Guessing and Compression*, Entropy **20**, 896 (2018).
13. W. Chen, Y. Cao, H. Wang, Y. Feng, *Minimum guesswork discrimination between quantum states*, Quantum Information & Computation **15**, 0737 (2015).
14. E. P. Hanson, V. Katariya, N. Datta, and M. M. Wilde, *Guesswork with Quantum Side Information*, IEEE Trans. Inform. Theory **68**, 322 (2022).
15. M. Dall'Arno, F. Buscemi, and T. Koshiba, *Guesswork of a quantum ensemble*, IEEE Trans. Inform. Theory **68**, 3193 (2022).
16. S. Sahni, T. Gonzalez, *P-Complete Approximation Problems*, Journal of the ACM **23**, 555 (1976).
17. T. C. Koopmans, M. Beckmann, *Assignment problems and the location of economic activities*, Econometrica **25**, 53 (1957).
18. E. Dragoti-Çela, *The quadratic assignment problem: Theory and algorithms* (Kluwer Academic Publishers, 1998, Dordrecht, The Netherlands).
19. R. E. Burkard, E. Dragoti-Çela, P. M. Pardalos, and L. S. Pitsoulis, *The Quadratic Assignment Problem*, Handbook of Combinatorial Optimization, Kluwer Academic Publishers, 241 (1998).
20. G. Laporte and H. Mercure, *Balancing hydraulic turbine runners: A quadratic assignment problem*, European Journal of Operational Research **35**, 378 (1988).
21. D. E. Knuth, *The art of computer programming*, Volume 4A, Pearson Education (2011).
22. R. Sedgewick, *Permutation generation methods*, Computing Surveys **9**, 137 (1977). Notice that in Algorithm 3a therein, the third line from the last should read "c[i] := 1; d[i] := not d[i]; i := i + 1;".
23. `https://codeberg.org/mda/libgw`
24. P. T. Highnam, *Optimal Algorithms for Finding the Symmetries of a Planar Point Set*, Carnegie-Mellon University (1985).
25. Jan D. Wolter, Tony C. Woo, and Richard A. Volz, *Optimal algorithms for symmetry detection in two and three dimensions*, The Visual Computer **1**, 37 (1985).
26. P. Brass and C. Knauer, *Testing the congruence of d-dimensional point sets*, International Journal of Computational Geometry and Applications **12**, 115 (2002).
27. H. Kim and G. Rote, *Congruence Testing of Point Sets in 4-Space*, Proc. 32nd International Symposium on Computational Geometry (SoCG 2016), 48:1-48:16, Leibniz International Proceedings in Informatics (LIPIcs) 51.
28. R. S. Bird, *A simple division-free algorithm for computing determinants*, Information Processing Letters **111**, 1072 (2011).