Quantum Machine Intelligence (2022) 4:24
https://doi.org/10.1007/s42484-022-00077-x

RESEARCH ARTICLE

Computing graph edit distance on quantum devices

Massimiliano Incudini[1] · Fabio Tarocco[1] · Riccardo Mengoni[2] · Alessandra Di Pierro[1] · Antonio Mandarino[3]

Received: 31 May 2022 / Accepted: 9 June 2022 / Published online: 26 August 2022
© The Author(s) 2022

Abstract
Distance measures provide the foundation for many popular algorithms in Machine Learning and Pattern Recognition. Different notions of distance can be used depending on the types of the data the algorithm is working on. For graph-shaped data, an important notion is the Graph Edit Distance (GED) that measures the degree of (dis)similarity between two graphs in terms of the operations needed to make them identical. As the complexity of computing GED is the same as NP-hard problems, it is reasonable to consider *approximate* solutions. In this paper, we present a QUBO formulation of the GED problem. This allows us to implement two different approaches, namely quantum annealing and variational quantum algorithms, that run on the two types of quantum hardware currently available: quantum annealer and gate-based quantum computer, respectively. Considering the current state of noisy intermediate-scale quantum computers, we base our study on proof-of-principle tests of their performance.

Keywords Graph edit distance · Quantum annealing · Variational quantum algorithms

1 Introduction

The *Graph Edit Distance* (GED) (Sanfeliu and Fu 1983; Gao et al. 2010) represents one of the most common dissimilarity measures used in Pattern Recognition and Image Processing (Conte et al. 2003; 2004). It has been successfully applied to many real-world tasks such as image recognition, handwritten digit recognition, face and face expression recognition (Filatov et al. 1995; Wiskott et al. 1997; Kisku et al. 2010; Mengoni et al. 2021; Neuhaus and Bunke 2004), and has applications in a variety of areas from computer vision and bioinformatics (Zaslavskiy 2010) to cognitive science (Mheich et al. 2015), and hardware security (Fyrbiak et al. 2020). It has also been used in Machine Learning in order to define more powerful kernels for support vector machines (Bellet et al. 2011), and in combination with kernel machines for pattern recognition (Neuhaus and Bunke 2007; Mengoni and Di Pierro 2019).

In general, the notion of edit distance (originally introduced for strings and then extended to graphs and general structured data) is given in terms of the operations that must be performed on a pair of data in order to make them identical, and represents a quantitative estimate of their dissimilarity. For example in pattern recognition it measures the strength of the distortions that have to be applied to transform a source pattern into a target pattern. As computing the edit distance is essentially to search for the best (in terms of cost) set of operations in the space of all possible ones, the problem is intrinsically a combinatorial optimization problem and its complexity depends on the structure of the data in the search space. When data are graphs, calculating the edit distance becomes easily inefficient as the complexity of the search space grows exponentially with the number of nodes of the graphs. In fact, the GED problem is an NP-hard optimization problem (Zeng et al. 2009), which makes exact approaches impossible to use for large graphs. This makes the study of approaches that give near-optimal results in reasonable time an urgent and essential one, also in the light of the practical impact of the GED problem. It is therefore of great importance to discuss and analyze in this context the feasibility of quantum approaches. Many classical

✉ Massimiliano Incudini
massimiliano.incudini@univr.it

✉ Antonio Mandarino
antonio.mandarino@ug.edu.pl

1 Dipartimento di Informatica, Università di Verona, Verona 34137, Italy

2 International Centre for Theory of Quantum Technologies, CINECA, Casalecchio di Reno 40033, Italy

3 International Centre for Theory of Quantum Technologies, University of Gdańsk, 80-309 Gdańsk, Poland

algorithms have been proposed up to now (Gao et al. 2010), which however cannot be straightforwardly translated to the quantum setting.

In this paper we introduce a formulation of the GED problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem, and show that this offers a common base for a direct implementation of the problem on both quantum and classical hardware, thus making it easy to compare the performances of various approaches. Our formulation is based on a *quantified* version of the graph isomorphism problem, where the expected answer is the actual number of operations that are needed to make two graphs isomorphic, rather than "yes" or "no".

The quest for optimization algorithms that can run on noisy intermediate-scale quantum (NISQ) computers is a very timely topic, as well as a challenging task (Preskill 2018; Alexeev et al. 2021; Martina et al. 2022). The two main quantum strategies that have been proposed to tackle optimization tasks are Quantum Annealing (QA) (Kadowaki and Nishimori 1998; Silva et al. 2020; Asproni et al. 2020) and the circuit-based variational approaches. In this paper we will assess the suitability of quantum annealing and two variational algorithms, i.e., Variational Quantum Eigensolver (VQE) (Peruzzo et al. 2014) and Quantum Approximate Optimization Algorithms (QAOA) (Farhi et al. 2014), on finding an approximate solution of the GED problem.

Moreover, we will give a concise comparative study of their performance with respect to a specific classical algorithm, namely Simulated Annealing, which makes use of the same QUBO formulation of the problem as the quantum strategies.

1.1 The quantum strategies

The QA approach can be seen as the quantum analogue of the classical simulated annealing optimization algorithm (Kirkpatrick et al. 1983), and essentially consists in using a specific search procedure for finding the state of minimum energy of spin systems studied in statistical mechanics, e.g., the Ising model in a random field (Kadowaki and Nishimori 1998). In the quantum case the fluctuations needed to scan the energy landscape are provided by a field that allows quantum tunneling, in contrast to the simulated annealing where the fluctuations are thermal (Hauke et al. 2020). While QA is based on the adiabatic model of quantum computation, VQE and QAOA are hybrid quantum-classical algorithms implemented on circuit-based quantum computers. This approach has been tested in several proof-of-principle experiments for real-world optimization (e.g., scheduling) problems (Vikstål et al. 2020; Bengtsson et al. 2020).

Both kinds of approaches share a common physical ground, since they are both implemented by means of a total Hamiltonian that is composed by two non-commuting terms, H_0 and H_1, such that $[H_0, H_1] = H_0 H_1 - H_1 H_0 \neq 0$. This Hamiltonian has the form

$$H = (1 - \lambda(t))H_0 + \lambda(t)H_1, \tag{1}$$

where $\lambda(t)$ is a control function that is valued in the interval [0, 1] and allows us to switch between the two terms. In fact, assuming that the computation time is in the interval $[0, \tau]$, one can initialize the system into the ground state of H_0, imposing $\lambda(0) = 0$, and tune the control parameter until reaching $\lambda(\tau) = 1$. If the variation of $\lambda(t)$ is such that the hypotheses of the adiabatic theorem (Kato 1950) are satisfied, both algorithms can be viewed as prefatory approaches for full adiabatic quantum computation (Albash and Lidar 2018). Moreover, the gates composing the QAOA can be seen as a Suzuki-Trotter decomposition (Mbeng et al. 2019) of the unitary evolution stemming from a Hamiltonian as defined in Eq. (1).

Another common aspect of QA and QAOA is that their performance significantly depends on heuristics for the choice of the annealing schedule and parameter initialization, respectively.

By defining the problem of calculating the GED as a Quadratic Unconstrained Binary Optimization (QUBO) problem, we are able to experiment both on quantum annealers, such as the D-Wave Systems Inc. machines, and on quantum circuits via variational algorithms such as VQE and QAOA. We introduce a QUBO formulation of GED that is inspired by the one presented in Calude et al. (2017) for the Graph Isomorphism problem, i.e., the computational problem of determining whether two finite graphs are isomorphic. Our formulation exploits the fact that GED can be seen as a quantitative generalization of graph isomorphism, which requires a "counting" phase while checking the nodes/edges in each of the graphs. This allows us to obtain a quantitative answer to the optimization problem rather than just a yes/no answer as in the case of graph isomorphism. This also explains why the GED problem is far more complex than graph isomorphism, for which it was recently shown that the problem is solvable in quasi-polynomial time (Babai 2016).

Our strategy is to use the QUBO formulation across all the platforms: classical, quantum annealers and gate-based quantum computers. This formulation of the problem, which is common for all the platforms, allows their smooth comparison and the identification of the most suitable one. The NISQ devices that we use to run the QUBO formulation of GED are the D-Wave System quantum annealer and the IBM circuit-based quantum computer. This will contribute to assess the actual power of NISQ devices for real-world problems. A benchmark of quantum optimization algorithms on satisfiability and max-cut problems has been introduced in Willsch et al. (2020). In contrast to that work, we analyze the GED problem, which requires a more involved QUBO

formulation and therefore a larger number of qubits. Moreover, we consider a broader class of quantum algorithms and include also classical algorithms in our comparison.

This paper is structured as follows. In Section 2 we introduce the notion of graph edit distance, and in Section 3 its QUBO formulation. Section 4 is devoted to the classical approaches developed to compute GED, while in Section 5 we discuss the methods we employ in our experiments on both quantum annealer and circuit-model quantum computer. In Section 6 we present the results of our experiments and provide a comparison among the different strategies. Finally, in Section 7 we discuss our results in the light of the currently available technologies and we address some open questions.

2 Graph edit distance

A graph $G = (V, E)$ consists of a finite, non-empty set of vertices V of cardinality $|V| = n$, and a set of edges $E = \{\{u, v\} \mid u, v \in V\} \subseteq V \times V$ of cardinality $|E| \leq n^2$. The graph is *undirected* if each edge is described by an unordered pair of vertices, *directed* otherwise. In an undirected graph, the number of edges is at most $n(n+1)/2$ or, by excluding *self-loops* (edges starting and ending in the same vertex), $n(n-1)/2$.

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic*, denoted by $G_1 \cong G_2$, if there exists a *bijection* (1-to-1 mapping) π between the vertex sets of G_1 and G_2 satisfying the property of *edge-preserving* (Fortin 1996), i.e., $\{u, v\} \in E_1 \Leftrightarrow \{\pi(u), \pi(v)\} \in E_2$. The *graph isomorphism* problem is the problem of establishing the exact matching of two graphs (Fortin 1996). The inexact matching is the more general case where there is a difference between two graphs, and a measure of this difference quantifies "how much" the two graphs are (dis)similar. An important graph similarity measure is the *graph edit distance* (GED) (Sanfeliu and Fu 1983; Gao et al. 2010), whose value is the total cost of "transforming" one graph into the other, thus making them isomorphic. Clearly, when two graphs are isomorphic, the GED between them is zero.

The GED problem is a NP-hard optimization problem, i.e., intuitively, it is at least as hard as the hardest problems in NP. This means that it is computationally more expensive than graph isomorphism for which a recent result shows that it is possible to find a solution in quasi-polynomial time (Babai 2016).

A *graph edit operation* is a mapping from the set of graphs to itself. The most common edit operations on a graph $G = (V, E)$ are listed in Table 1, where we also specify the part of the graph which they act on (Vertex or Edge set), and the condition of their applicability (Constraint). Specifically, these operations can be performed provided that they do not insert a pre-existing vertex/edge, or delete a non-existing vertex/edge; it must also be guaranteed that before deleting a vertex, each edge starting or ending into that vertex must be deleted.

Table 1 Main operations composing a graph edit path

Operation	Vertex set	Edge set	Constraint
Insert node v	$V \cup \{v\}$	No changes	$v \notin V$
Delete node v	$V \setminus \{v\}$	No changes	$v \in V, \nexists uv \in E$
Insert edge uv	No changes	$E \cup \{uv\}$	$uv \notin E$
Delete edge uv	No changes	$E \setminus \{uv\}$	$uv \in E$

A *graph edit path* P is a composition of graph edit operations and the number of these operations defines the length, $\ell(P)$, of the path. The *graph edit distance* between two graphs G_1 and G_2 can be defined as

$$GED(G_1, G_2) = \min_P \{\ell(P) \mid P(G_1) \cong G_2\}. \tag{2}$$

A more fine-grained definition of GED can be found in Blumenthal and Gamper (2020), where *labelled graphs* are considered, so that each graph edit operation might contribute to the graph edit distance with a different weight. For our purpose, it is sufficient to consider the case of unlabelled graphs, which allows us to keep our implementation simpler. In particular, in the calculation of the GED we will assume that each edit operation has cost 1. This implies that the length of an edit path effectively corresponds to the number of operations composing it, which is a reasonable estimate of the complexity of the calculation. An example of calculation of GED is shown in Fig. 1.

3 QUBO formulation of GED

The graph matching problem can be encoded as a linear optimization problem (Almohamad and Duffuaa 1993; Justice and Hero 2006) or a quadratic optimization problem (Vogelstein et al. 2015; Calude et al. 2017). We show here how the second approach can be extended to inexact graph matching, and precisely to the GED problem. The idea is to reformulate GED as a Quadratic Unconstrained Binary Optimization (QUBO) problem, a class of problems that is well known in multivariate optimization. As the name suggests, a QUBO problem is defined in terms of a *quadratic* function of *binary* variables x_i, which is *unconstrained* or, more precisely, with constraints replaced by penalty terms and encoded within a matrix Q representing the objective function (Kochenberger et al. 2014). The task is to find the value $\mathbf{x}^*$ such that

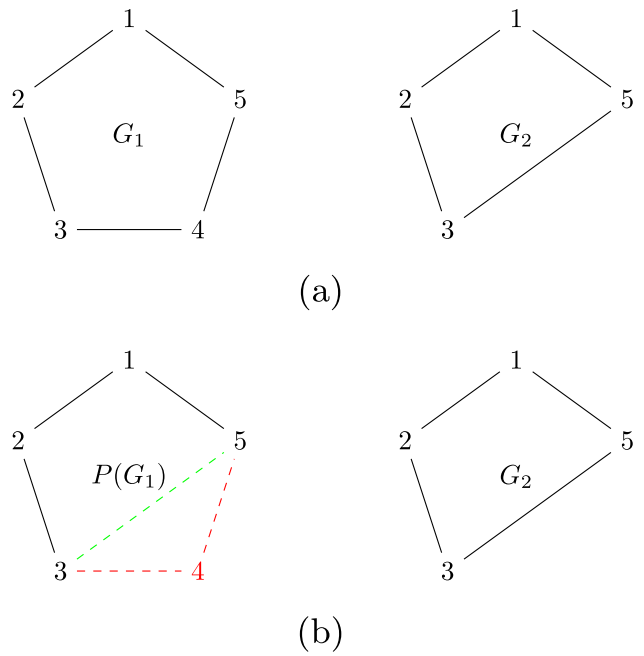

Fig. 1 Solution of an instance of GED problem. (**a**) Initial graphs. (**b**) The edit path P consisting of four edit operation (insertion of {3, 5}, deletion of {3, 4},{4, 5}, deletion of vertex 4) is such that $P(G_1) \cong G_2$. Moreover $\ell(P)$ is the GED since no shorter path exists that makes the graphs isomorphic

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathbf{x}^T Q \mathbf{x}, \tag{3}$$

where $x_i \in \{0, 1\}$ for any of the n entries of $\mathbf{x}$, and Q is an $n \times n$ upper triangular real valued matrix with elements $q_{i,j}$ encoding the data specification.

QUBO problems are NP-hard problems (Pardalos and Jha 1992) and share a similar structure and the same computational complexity with spin glass Ising models. This similarity allows for an almost immediate implementation of QUBO objective functions into the adiabatic model of quantum computation.

A QUBO formulation of the GED problem can be given as follows. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ we shall assume that the number of vertices is the same in both graphs, that is $|V_1| = |V_2| = n$. This is without loss of generality; in fact, if G_1 has $k = |V_1| - |V_2| > 0$ vertices more than G_2, we build an auxiliary graph G_2' by adding k isolated vertices to G_2; then, we calculate he GED between G_1 and G_2', and finally add k to the given value for considering the k insertion node edit operations. The formulation requires n^2 variables, each denoted by $x_{i,j}$, with $x_{i,j} = 1$ if and only if the i th vertex of G_1 is mapped to the j th vertex of G_2.

We define the dual path of P as the path, $\tilde{P}$, where any deletion operation becomes an insertion operation and vice versa. If a bijection π corresponds to a graph edit path P such

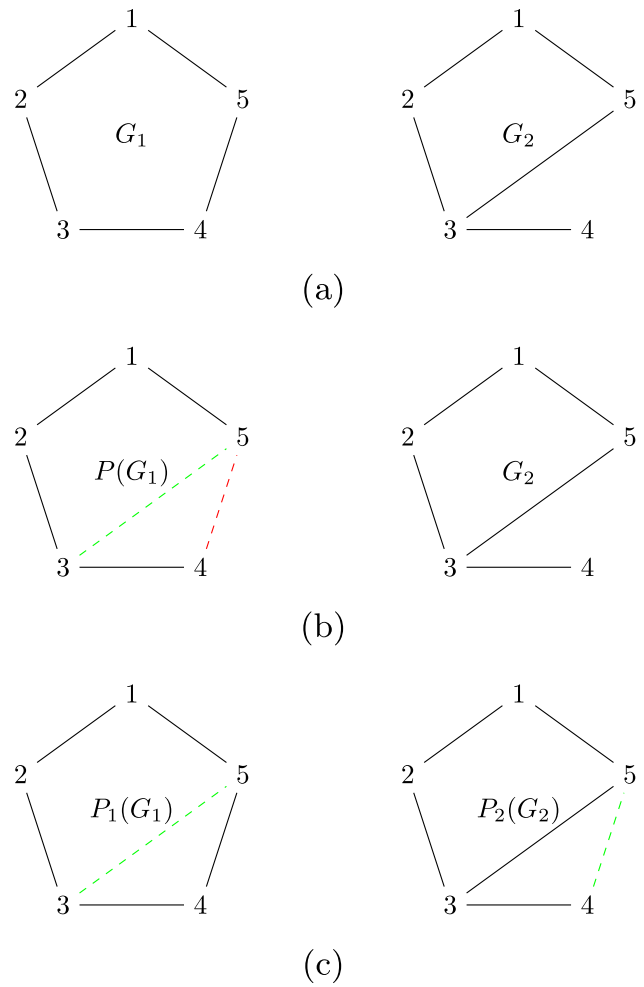

Fig. 2 Equivalence between the standard and the insertion-only definition of GED between G_1 and G_2, with GED=2. (**a**) The bijection used to calculate the GED, $\pi(i) = i \forall i \in [1,5]$. (**b**) A graph edit path $P, |P| = 2$, such that $P(G_1) \cong G_2$ according to π. The insertion is denoted by a green dashed line and the deletion by a red dashed line. (**c**) Paths P_1 and P_2 such that $|P_1| + |P_2| = 2$ and $P_1(G_1) \cong P_2(G_2)$ according to π. The only insertion operations are denoted by a green dashed line

that $P(G_1) \cong G_2$, we can partition P in two edit paths P_1 and P_2, where P_1 has only insertion operations and P_2 only deletion operations. Then

$$P(G_1) \cong G_2 \text{ if and only if } P_1(G_1) \cong \tilde{P}_2(G_2).$$

Clearly, $\ell(P_1) + \ell(P_2) = \ell(P_1) + \ell(\tilde{P}_2) = \ell(P)$. Intuitively, the process is illustrated by the example reported in Fig. 2.

Considering the two assumption stated above, we can define the cost of a bijection π as

$$\mathbf{cost}(\pi) = |E_1 \setminus \pi^{-1}(E_2)| + |E_2 \setminus \pi(E_1)| \tag{4}$$

where $\pi(E) = \{\{\pi(i), \pi(j)\} \mid \{i, j\} \in E\}$. For a given bijection π, the value of cost (π) is equal to the number of edges occurring in E_1 and not in E_2 plus the number of edges present in

E_2 but not in E_1. The assumption that the graphs have the same number of vertices guarantees that a bijection exists.

We can now define the GED as:

$$GED(G_1, G_2) = \min_{\pi} \mathbf{cost}(\pi). \tag{5}$$

The QUBO formulation of our problem is composed by two parts: a *hard constraint* Q_h, whose function is to add a penalty if the solution $\boldsymbol{x}$ does not represent a bijection (in that case, $\boldsymbol{x}$ is not a solution for GED) and a *soft constraint* Q_s, which introduces a penalty for any edge mismatch as in Eq. (4).

The hard constraint reads as

$$\begin{aligned} Q_h(\boldsymbol{x}) = & \sum_{0\le i<n} \underbrace{\left(1 - \sum_{0\le j<n} x_{i,j}\right)^2}_{(a)} \\ & + \sum_{0\le j<n} \underbrace{\left(1 - \sum_{0\le i<n} x_{i,j}\right)^2}_{(b)} \end{aligned} \tag{6}$$

and ensures the validity of the solution (i.e., $\boldsymbol{x}$ corresponds to a bijection). Whenever the i th vertex of G_1 is mapped to zero or more than one vertices of G_2, the (a) term of Eq. (6) is greater than zero. The (b) term of Eq. (6) is greater than zero, when the j th vertex of G_2 is the image of none or more than one vertices of G_1.

The soft constraint is

$$\begin{aligned} Q_s(\boldsymbol{x}) = & \sum_{\{i,j\}\in E_1} R_{i,j}(\boldsymbol{x}) \\ & + \sum_{\{i',j'\}\in E_2} S_{i',j'}(\boldsymbol{x}) \end{aligned} \tag{7}$$

with

$$R_{i,j}(\boldsymbol{x}) = \sum_{0\le i'<n} x_{i,i'} \sum_{0\le j'<n} x_{j,j'}(1 - e^{(2)}_{i',j'})$$

and

$$S_{i',j'}(\boldsymbol{x}) = \sum_{0\le i<n} x_{i,i'} \sum_{0\le j<n} x_{j,j'}(1 - e^{(1)}_{i,j})$$

where

$$e^{(k)}_{i,j} = \begin{cases} 1, \{i,j\} \in E_k \\ 0, \{i,j\} \notin E_k \end{cases}.$$

The term Q_s counts how many edges are not preserved by the bijection π implied by $\boldsymbol{x}$. In particular, the $R_{i,j}$ counts the arcs $\{i,j\}$ in G_1, $\{\pi(i), \pi(j)\}$ missing in G_2. The $S_{i',j'}$ terms counts the arcs $\{i', j'\}$ in G_2, $\{\pi^{-1}(i'), \pi^{-1}(j')\}$ missing in G_1. We prove now that the term $\sum_{i,j\in E_1} R_{i,j}(\boldsymbol{x})$ is equivalent to the term $|E_1 \setminus \pi^{-1}(E_2)|$:

$$\begin{aligned} & |E_1 \setminus \pi^{-1}(E_2)| \\ = & \sum_{\{i,j\}\in E_1} (1 - e_{\pi(i),\pi(j)}) \\ = & \sum_{\{i,j\}\in E_1} \left(x_{i,\pi(i)} x_{j,\pi(j)} (1 - e_{\pi(i),\pi(j)})\right) \\ = & \sum_{\{i,j\}\in E_1} \left(\sum_{i'\le n} x_{i,i'} \sum_{j'\le n} x_{j,j'}(1 - e_{i'j'})\right) \\ = & \sum_{\{i,j\}\in E_1} R_{i,j}(\boldsymbol{x}) \end{aligned}$$

where e_{ij} is one iff arc $\{i, j\}\in E_2$. The procedure to identify $S_{i',j'}$ is equivalent. The complete formulation reads:

$$Q_{\lambda_h,\lambda_s}(\boldsymbol{x}) = \lambda_h Q_h(\boldsymbol{x}) + \lambda_s Q_s(\boldsymbol{x}) \tag{8}$$

where the choice of parameter λ_h, λ_s decides the weight of each penalty term.

If we set $\lambda_h \gg \lambda_s$ we are guaranteed that all *valid* solutions, i.e., all those having null hard constraint contribution, have lower cost than any *non-valid* solutions. For graphs of n vertices, we can choose

$$\lambda_h > n^2 \lambda_s, \tag{9}$$

to ensure that the contribution of the soft constraint $\lambda_s Q_s(\boldsymbol{x})$ to the total QUBO problem is always smaller than the contribution of hard one $\lambda_h Q_h(\boldsymbol{x})$. This is because n^2 is the cost of the worst case, i.e., when one graph is complete and the other one empty.

If the maximum cardinality of edges is $|E| = m < n^2$, then Eq. (9) becomes $\lambda_h > m\,\lambda_s$.

4 Classical and quantum approaches to GED

Many classical algorithms for the GED problem exist (Blumenthal and Gamper 2020; Blumenthal et al. 2020). Clearly, given the computational complexity of GED, all known exact algorithms run in exponential time. A widely used approach is based on the A* search algorithm (Abu-Aisheh et al. 2015). A different approach is to consider classical heuristics that run in polynomial time in the size of the input; however, they are not all suitable to treat QUBO problems. The two most common approaches are either based on reduction GED problem to LSAPE (Linear Sum Assignment Problem with Error-Correction) (Bougleux et al. 2017) or based on local search (Hoos and Tsang 2006).

A different approach is the Simulated Annealing (SA) heuristic (Kirkpatrick et al. 1983). This is a probabilistic algorithm that minimizes multivariate binary objective functions and can be used to solve QUBO problems. Due to its physical background discussed later in Section 5, it can be compared with the approaches presented in the following sections. SA can be seen as taking a random walk in the solution space according to a Markov chain parametrized

by a temperature parameter T (McGeoch 2014). We choose SA as classical counterpart in our comparison because it can straightforwardly be used to solve QUBO problems.

4.1 Simulated annealing

SA works as shown in Algorithm 2 in Appendix B. When starting the algorithm the temperature T is high, and solutions with higher energy are accepted with a probability that follows the Boltzmann distribution. The temperature decreases exponentially and SA has fewer chances to accept high energy solutions. When the temperature is zero, SA works in a gradient-descent fashion and will converge to a local minima. SA is restarted many times (shots), each from a different point of the state space chosen randomly. Finally, the energy of the solution is the single, lowest energy found in all the shots.

4.2 Quantum annealing

Quantum Annealing (QA) (Apolloni et al. 1989; Kadowaki and Nishimori 1998) is a meta-heuristic search algorithm that can be used to tackle QUBO problems. This optimization technique finds its roots in a problem mutuated by statistical physics, namely the search of the minimum-energy state of a spin system exhibiting a glassy phase (Mézard et al. 1987).

We briefly recall the notation and the physics underlying a class of spin systems, hereafter referred to as Ising-like model. The aim is to clarify the connection between the unconstrained quadratic problems and the search of the ground state of such a class of models.

Let us introduce a classical spin variable s_i that can take values ± 1. The function describing the energy of a system of N interacting spin disposed over a d −dimensional discrete lattice is the Hamiltonian:

$$H(s) = -\sum_{\langle i,j \rangle} J_{i,j} s_i s_j - \sum_i h_i s_i \tag{10}$$

where the $\langle i, j \rangle$ denotes that the sum is over all the first neighboring sites, the $J_{i,j}$ are the couplings between two sites of the lattice and h_i is the external magnetic field acting on each spin.

A quantum version of the Hamiltonian in Eq. (10) is obtained replacing suitably the binary variable s_i with an ad hoc Pauli matrix σ_i^α with $\alpha = \{x, y, z\}$:

$$H(\sigma) = -\sum_{\langle i,j \rangle} J_{i,j} \sigma_i^z \sigma_j^z - \sum_{i=1}^{N} h_i \sigma^x. \tag{11}$$

The problem of finding the ground state of a Hamiltonian describing an *Ising Spin Glass* problem is NP-hard and how it relates to the solution of many NP-hard is reported in Lucas (2014).

The QUBO problem is closely related to the problem of finding the ground state of a Hamiltonian written in terms of spin variable, upon introducing the transformation:

$$x_i \leftrightarrow \frac{1 + \sigma_i^z}{2} \tag{12}$$

Originally, QA was introduced in Apolloni et al. (1989) as a quantum-inspired, classical algorithm. In contrast to the SA in which the fluctuations to explore the energy landscape are provided by the temperature parameter T, quantum annealing uses a transverse field coefficient $\lambda(t)$ called *tunneling coefficient* that modulates the two terms of the Hamiltonian as in Eq. (1).

The quantum annealing uses the term H_0 usually referred to as the *transverse field Hamiltonian* that does not commute with the term H_1 in which the optimization problem is encoded. The non-commutativity of the two terms provides the fluctuations necessary to exploit the quantum tunneling, and allows us to escape from the local minima by tunneling through hills in the solution landscape (McGeoch 2014). Typically, when the quantum hardware is used, the system is in a superposition of all possible state, with probability amplitudes depending on $H(t)$.

Quantum annealers usually have a certain number of qubits which are connected according to a given topology. Logically, any QUBO variable is mapped to a qubit. Physically, it is possible that two variables linked by a quadratic term $J_{i,j}$ are not connected. This requires us to find a *minor embedding* (Choi 2008), thus a mapping of variables to physical qubits such that variables bound by quadratic terms are located to adjacent qubits. If this is not possible, additional qubits are required to represent a single variable. For this reason, to each variable is assigned one logical qubit but this can be mapped to more than one physical qubit. The task of finding a minor embedding consists in searching a minor of the graph associated with the hardware topology which is isomorphic to the one associated with the QUBO problem. Minor embedding is currently solved using heuristics, available on the D-Wave Ocean library.

It may happen that the physical qubits associated with a single logical qubit are prepared in different states: some in $s = -1$ and others in $s = +1$. This phenomenon is called *chain break* and is resolved in post-processing using either *majority vote technique* (the logical qubit has the most frequent value in the physical qubits assignment) or *energy minimization* (the logical qubit has the value that minimize the energy). The latter requires multiple evaluation of the formulation).

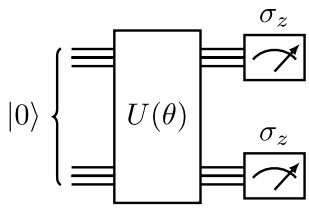

Fig. 3 Parametric quantum circuit

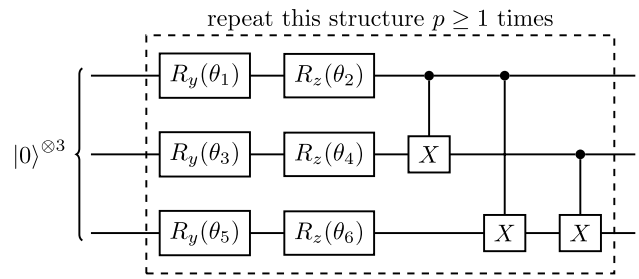

Fig. 4 Circuit for the variational ansatz used for VQE algorithm defined on $n = 3$ qubits. The structure can be repeated p times and the number of parameters θ is $2pn$

4.3 Variational quantum algorithms

We can use the QUBO formulation to solve the GED problem also on gate-based quantum computers, via Variational Hybrid algorithms, which are based on both classical and quantum resources.

We define a parametric quantum circuit (PQC), that depends on an *ansatz* for the values of the parameters $\boldsymbol{\theta} = (\theta_1, ..., \theta_M)$ (Fig. 3). The number of parameters M depends on the architecture of the circuit and the number of qubits n (which are equivalent to the variables of the QUBO problem). For example, we can choose rotational gates which naturally depends on a set of rotation angles. As a shorthand we denote the composition of unitary gates composing the PQC by $U(\boldsymbol{\theta})$, and the final state of such a circuit by $U(\boldsymbol{\theta})|0\rangle^{\otimes n} = |\Psi(\boldsymbol{\theta})\rangle$. Then we compute the expectation value of the problem Hamiltonian H_C:

$$E(\boldsymbol{\theta}) = \langle \Psi(\boldsymbol{\theta}) | H_C | \Psi(\boldsymbol{\theta}) \rangle. \tag{13}$$

Such Hamiltonian is related to the QUBO formulation by Eq. (12). The value $E(\boldsymbol{\theta})$ corresponds to the cost function which has to be minimized and the minimization stage is performed classically.

Examples of this kind of algorithms are the variational quantum eigensolver (VQE) (Peruzzo et al. 2014) and the quantum approximate optimization algorithm (QAOA) (Farhi et al. 2014). They are mostly used to find the ground state of the Hamiltonian of nonintegrable spin systems (McClean et al. 2016), which is indeed a minimization task.

The Variational Quantum Eigensolver (VQE) is inspired by the Variational Principle (Peruzzo et al. 2014; Wilson et al. 2021) and it has found its most groundbreaking application in chemical-physics simulation (Kandala et al. 2017; Chiesa et al. 2019; Crippa et al. 2021; Huembeli et al. 2022). The structure of the variational algorithm depends on the choice of an ansatz. Here, we consider the ansatz `EfficientSU2`, which sufficiently approximate the solution space as shown in Funcke et al. (2021).

The circuit implementing it is shown in Fig. 4, and its construction is detailed in Algorithm 3 in Appendix A.

The Quantum Approximate Optimization Algorithm (QAOA) recently introduced in Farhi et al. (2014), Vikstål et al. (2020), Bengtsson et al. (2020), and Zhou et al. (2020) is an application of VQE. The ansatz must respect a particular structure which depends on the Hamiltonian defining the problem.

5 Experimental setup

We consider a set of graphs with a number of vertices ranging from three to nine, and we randomly generate the graphs following the procedure explained in Batagelj and Brandes (2005). The procedure needs as inputs the number of vertices n and the probabilities p of generating edges in the graph. For all the configurations labelled by the number of vertices $n \in \{3, 4, ..., 9\}$, we generate four graphs G_1, G_2, G_3, G_4 with edge probabilities $p_1 = 0.1$, $p_2 = 0.33$, $p_3 = 0.66$, $p_4 = 0.99$. This means that graph G_1, generated with probability p_1, will contain a small number of edges while the graph G_4 is almost always a complete graph.

We proceed now to exactly compute the GED (G_i, G_j) with $i, j = 1, 2, 3, 4$ for each number n of vertices by means of the A* algorithm (Abu-Aisheh et al. 2015). Note that this is feasible only for small-sized graphs due to the exponential worst-case complexity of the algorithm. Any computational run has four inputs: the first graph G_i, the second graph G_j and the parameters λ_h, λ_s of Eq. (8). We iterate the runs over all possible pairs of graphs with the same number of vertices, including the pair of a graph with itself.

The formulation shown in Eq. (8) requires to set some values for parameters λ_h, $\lambda_s > 0$. Experimental results, obtained by running SA on all pairs of values $\lambda_h = 1$ and $\lambda_s \in \{1/i | i = 1, ..., 10\} \cup \{0.05, 0.01\}$ on graphs up to 9 vertices, show that the actual best value for λ_s fulfilling Eq. (8) would be $\lambda_s < \lambda_h/81 \approx 0.012$. Our choice of setting $\lambda_h = 1$ and $\lambda_s = 0.1$ is a compromise between the need to fulfill Condition (9) — which is valid for small λ_s — and the need to avoid

Table 2 Average relative difference for the experiments with $\lambda_h = 1$, $\lambda_s \in \{1/i | i = 1,..., 10\} \cup \{0.05, 0.01\}$ with respect to the number of vertices n

	Values of λ_s											
n	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	0.1	0.05	0.01
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.52	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
6	0.53	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.54	0.32	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.09
8	0.48	0.48	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.15
9	0.62	0.50	0.38	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.09	0.24

values, $\lambda_s \simeq 0$, which are so low as to make the soft constraint irrelevant.

We have run SA 1000 times, and kept the lowest energy solution $\tilde{\boldsymbol{x}}$. For each run we have calculated the exact GED s and the approximated one $\tilde{s}$. As the *absolute error* $|s - \tilde{s}|$, grows with the problem size, we have introduced another quantity called the *relative difference*, which is defined as

$$\Delta = \begin{cases} 0, & s = \tilde{s} \\ \left| \frac{\tilde{s} - s}{\max\{s, \tilde{s}\}} \right|, & s \neq \tilde{s} \end{cases} \tag{14}$$

and lies in the range [0, 1]. The results of our experiments are shown in Table 2. They suggest that the best values are those with $\lambda_s \in [1/4, 1/10]$. Choosing values of λ_h and λ_s, such that the condition (9) is satisfied, ensures the existence of a valid solution; however, this is only a sufficient condition. In fact, we have strong numerical evidences that valid solutions exist also for other choices of parameters, for which a constraint linking λ_h and λ_s is still an open question. It is arguable that the best choice of parameters for SA will not necessary be the best one for quantum annealers and QAOA. However, notice that this choice is guided by the considerations explained above. The results of the experiments shown in Table 2 empirically show the effectiveness of our choice.

The experimental setup is based on state-of-the-art, freely available software. This approach facilitates reproducibility and avoids reinventing the wheel issues, e.g., bugs, non-standard implementations. As argued in Mauerer and Scherzinger (2022), reproducibility is a serious issue in quantum science. We have addressed the problem by releasing the code in github.com/incud/GraphEditDistance. However, quantum annealer-based experiments require the access to D-Wave machines which is available through the cloud.

5.1 Classical approaches

The exact calculation of GED is performed through the A* algorithm (Abu-Aisheh et al. 2015) implemented in NetworkX (Hagberg et al. 2008), while as heuristic we use the SA implementation provided by D-Wave Ocean SDK.

The control parameters in this implementation are set as follows. The temperature is initialized at $T_{start} = M/\log 2$ and decreases exponentially to $T_{end} = m/\log 100$, where M and m are an upper bound and a lower bound of the solutions, respectively. The (absolute value of the) smallest entry of the QUBO matrix is the lower bound m. The sum of the (absolute value of the) entries of the QUBO matrix is the upper bound M.

Another implementation of SA is the one provided within the GEDLib library (Blumenthal et al. 2019) and its Python wrapper GEDLibPy [1]. The comparison between this and the one provided by D-Wave is reported in Appendix B and shows that the results are comparable. We have also tested other state-of-the-art heuristics that are not based on QUBO formulation; however, in general SA is the best performing (see Appendix B). However, we will rule out these heuristics from our comparison with the quantum algorithms and restrict ourselves only to methods specifically designed for QUBO problems.

5.2 D-Wave quantum annealers

For our experiments we have used three different, (fully) quantum annealers from hardware company D-Wave: (i) D-Wave 2000Q; (ii) D-Wave Advantage 1.1; (iii) D-Wave Leap Hybrid Solver. We recall briefly their different characteristics.

D-Wave 2000Q has 2041 qubits and uses Chimera topology having 5974 couplers (physical connections between qubits) (D-Wave Systems Inc 2020a). D-Wave Advantage 1.1 has 5436 qubits and uses a Pegasus topology having 37440 couplers (D-Wave Systems Inc 2020b). Chain breaks are resolved using *majority vote*, which is the cheapest technique and is the one suggested by the hardware vendor. D-Wave Leap Hybrid Solver is not a quantum annealer but a hybrid classical-quantum software whose configurations are fully managed by D-Wave and it is not customizable.

[1] https://github.com/Ryurin/gedlibpy

Both D-Wave 2000Q and Advantage allow us to set the configuration of the annealing process, controlled by a parameter s that grows monotonically from 0 to 1. The parameter is defined as $s = t/\tau$, where τ is the annealing time. For a specified *annealing time* the evolution proceeds linearly in t.

Due to the enhanced topology, it is expected to find more compact embedding with the latter hardware (McGeoch and Farré 2020).

Any GED computation runs on both machines with the following configurations:

1. *default configuration (DC)*: the annealing time is 20 µs, the annealing process proceeds linearly;
2. *configuration Short Time (ST)*: the annealing time is 1 µs, the annealing process proceeds linearly;
3. *configuration Long Time (LT)*: the annealing time 500 µs, the annealing process proceeds linearly;
4. *configuration Pause Middle (PM)*: annealing schedule is

   (a) in the first 5 µs, s grows from 0 to 0.45,
   (b) in the following 94 µs, s remains constant,
   (c) in the last 1 µs, s grows from 0.45 to 1.

   According to Marshall et al. (2019), the performance of pausing within the annealing schedule depends uniquely on the length of the pause (longer is better) and are independent of the annealing process before and after the pause.

A run is the number of times the annealing is restarted. We have tried each configuration with a different number of runs: ST with 10^4, DC with both 10^3 and 10^4, the others with 10^3 and we have kept only the lowest energy result.

We expect that the long time configuration might outperform those having shorter annealing time, due to the possibility of exploring the energy landscape widely. However, for longer times decoherence effects due to noise may arise and there is still no direct control of it. This increases the probability of errors.

We also expect the PM configuration to outperform the default configuration, due to the previous evidences suggesting that pausing the schedule in the middle of the process improves the performances (Marshall et al. 2019; Passarelli et al. 2019; Chen and Lidar 2020).

5.3 Variational quantum algorithms

We used the implementation of VQE and QAOA available on the Qiskit IBM platform (IBM 2021).

We briefly recall the construction of the circuit implementing the QAOA as in Farhi et al. (2014) and in Fig. 5 we plot the circuit used to tackle the GED problem. The circuit has as input the state $|+\rangle^{\otimes n}$ where n is the number of variables of the QUBO problem. Such state is obtained from $|0\rangle^{\otimes n}$ by applying the Hadamard gate to each qubit. The QAOA ansatz is constructed by repeating p times two unitary operation $U_C(\boldsymbol{\gamma})$, $U_B(\boldsymbol{\beta}_s)$. The whole ansatz will depend on $2p$ parameters $\gamma_1, \beta_1, \ldots, \gamma_{2p}, \beta_{2p} \in [-\pi, \pi]$. The first unitary operator is defined to be $U_C(\gamma) = \exp\{-i\gamma H_C\}$ where H_C is the problem Hamiltonian and depends on our input. The mapping between the QUBO formulation and the Hamiltonian formulation is detailed in Eq. (12). The second unitary operator is called *mixing Hamiltonian* and is defined to be $U_B(\beta) = \exp\{-i\beta H_B\}$, where $H_B = \sum_i \sigma_i^x$. Note that the two Hamiltonians H_C, H_B must not commute to have not trivial results. The circuit gives as output the state

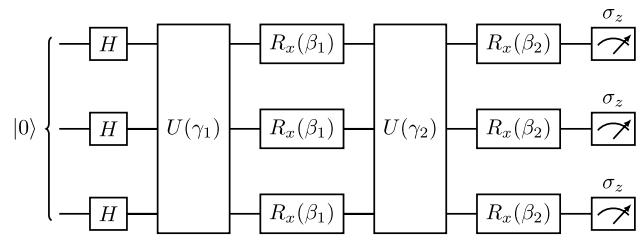

Fig. 5 Scheme of the QAOA circuit for $n = 3$ qubits. The variational ansatz is repeated twice. The number of parameters is fixed to $2p$ (and does *not* depends on n)

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U_B(\beta_p)U_C(\gamma_p)\cdots U_B(\beta_1)U_C(\gamma_1)|+\rangle^{\otimes n}. \tag{15}$$

The expectation value represents the energy (cost) associated with a particular choice of parameters and must be minimized:

$$E(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|H_C|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle, \tag{16}$$

and we stress that the above equation is Eq. (13) rewritten for the QAOA case. Finally, the task of the classical optimization is to find the optimal variational parameters such that:

$$(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*) = \arg\min_{\boldsymbol{\gamma}, \boldsymbol{\beta}} E(\boldsymbol{\gamma}.\boldsymbol{\beta}) \tag{17}$$

We perform 2048 runs with randomly initialized parameters, then the classical optimization is performed using the algorithm COBYLA (Powell 1994). [2] The construction of QAOA ansatz is detailed in Algorithm 4 in Appendix A.

[2] There are evidences that in the statevector simulation COBYLA slightly outperforms the other optimizers (Pellow-Jarman et al. 2021).

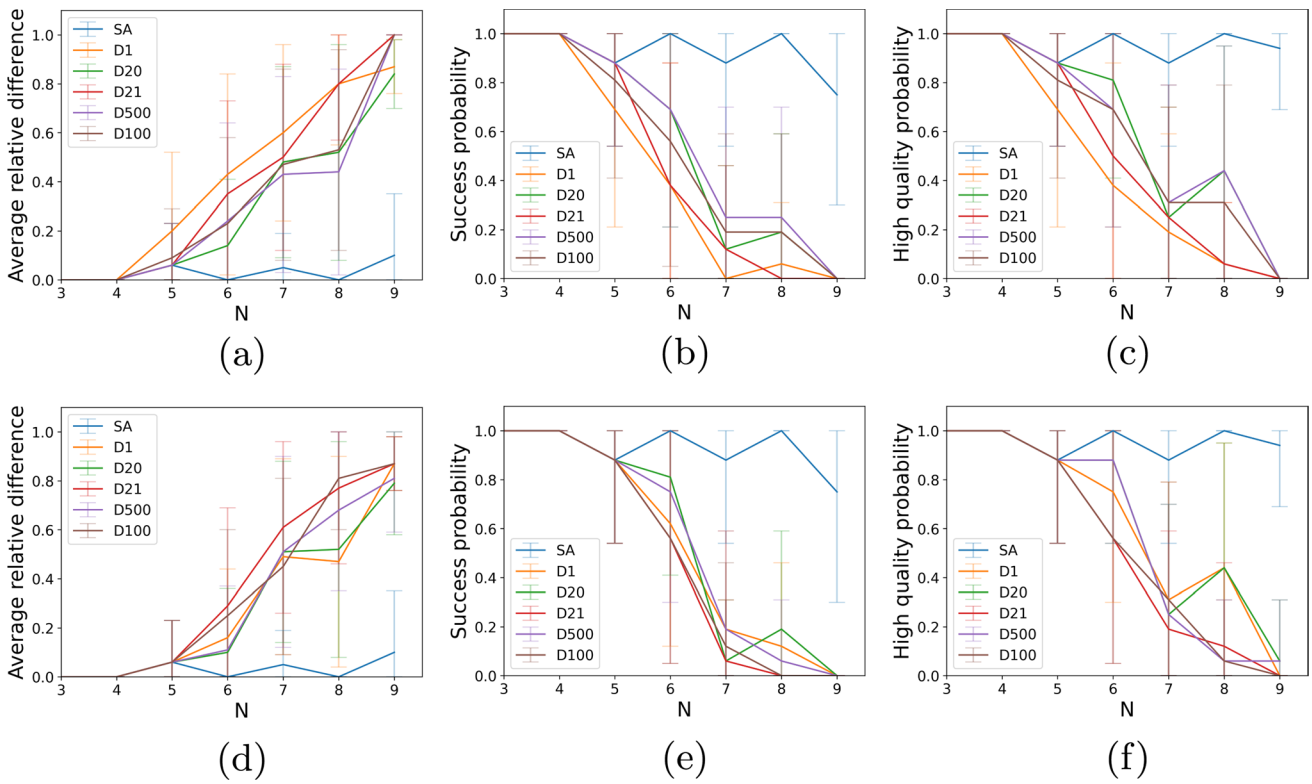

Fig. 6 Comparison of the performance of the many configurations of quantum annealers as function of n the number of vertices of a pair of graphs. Plots (**a**)-(**b**)-(**c**) compare configurations of D-Wave 2000Q, while (**d**)-(**e**)-(**f**) compares Advantage hardware. Plots (**a**)–(**d**) show the relative difference, (**b**)–(**e**) the success probability, (**c**)–(**f**) the high quality probability. The error bars represent the standard deviation. Legend: SA is simulateed annealing, D1 is annealing 1μs × 10^4 runs, D20 is annealing 20μs × 10^4 runs, D21 is annealing 20μs × 10^3 runs, D500 is annealing 500μs × 10^3 runs, D100 is annealing 100μs paused in the middle and × 10^3 runs

6 Numerical results

Our analysis takes into account the number of vertices n, the hardware and its configurations. We evaluate the performance of each algorithm, illustrated in Appendix A, using three different measures: the average of the *relative difference* Δ, the average of *success probability*, and the average of *high-quality probability*. The relative difference was defined in Eq. (14); the success probability is defined as the percentage of experimental results having $\Delta = 0$, and the *high-quality probability* is defined as the percentage of results having $\Delta \leq 0.2$.

All experiments uses the QUBO formulation shown in Eq. (8) with parameters $\lambda_h = 1$, $\lambda_s = 0.1$, which are optimal according to the preliminary analysis in Table 2. The choice of these values enforces the hard constraint, maximizing the chances of reaching a valid solution.

We summarized our results in Figs. 6, 7 and 8. All the numerical results are reported in Appendix B.

6.1 Resource usage

The resources exploited by the quantum annealer can give us useful information about the performance of our optimization tasks implemented over different configurations. We focus on the number of *logical qubits*, i.e., the number of variables of the problem, and *physical qubits*, i.e., the number of qubits needed to encode the problem by means of the minor embedding procedure. We consider the *maximum length of the chain*, that is the maximum number of physical qubits needed to encode a single logical qubit, and *the chain strength* that measures the strength of the interaction between physical qubits belonging to the same chain.

As shown in Table 3, the number of logical qubits depends uniquely on the graph size. The other resources depend on both the topology of the quantum annealer and the quality of the software performing the minor embedding (which is the same for both versions of the hardware). It is evident that D-Wave Advantage produces much smaller

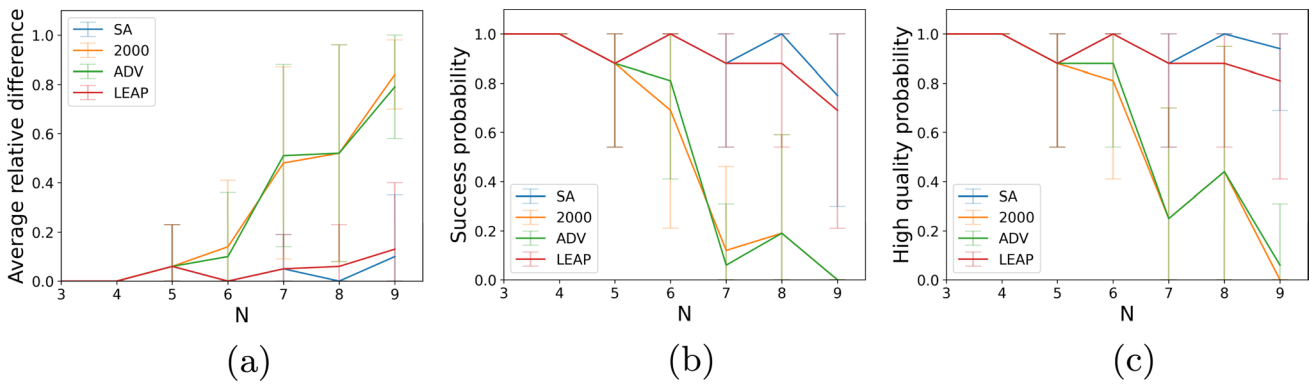

Fig. 7 Comparison of the performance of SA and the different versions of quantum annealers. (**a**) shows the average relative difference, (**b**) shows the success probability, (**c**) shows the high-quality probability. The error bars represent the standard deviation. Legend: SA is simulated annealing, D2000 is D-Wave 2000Q having annealing time 20μs × 10^4 runs, ADV is D-Wave Advantage 1.1 having annealing time 20μs × 10^4 runs, LEAP is D-Wave Leap

embedding; as shorter chains lead to fewer errors, and in fact we obtain more accurate solutions.

Then, we have quantified the resources required for variational algorithms. In Table 4 we report the *number of qubits* which is exactly to the number of variables. Since there is no need of minor embedding, the number of physical qubits corresponds to the number of logical ones. We also report the *number of parameters* to be trained by the classical optimizer, the *depth*, i.e., the maximum number of gates on one qubit, and the *size*, i.e., the total number of gates (see Childs et al. 2019 for a more formal definition of the notions of size and depth of a circuit).

To have a good estimation of resources, we transpile the circuit in terms of single-qubits rotations U_3 and CNOT gates.

We immediately see that in general VQE requires much more parameters than QAOA, although it depends mostly on the choice of the variational form. In general, this should lead to longer classical optimization phase.

We were able to run experiments with graphs up to 5 vertices, since larger instance requires much more computational power for the simulation, which was not available to us. The number of required gates suggests that this approach is not feasible on NISQ hardware due to the low gate fidelity and decoherence errors. Thus, we have performed the calculation on error-free simulators.

6.2 Comparing quantum annealers

We identify the best performing configuration for all the quantum annealers, and then compare their performance with the simulated annealer. Figure 6 compares the different configuration of D-Wave 2000Q and Advantage 1.1. For both quantum annealer versions we see that:

- configurations having 10000 runs perform significantly better than configurations with 1000 runs;
- the annealing time that minimizes the relative difference is $\tau \geq 20$μs. Shorter values return inaccurate solutions, and larger values have similar performance but are costly.
- the introduction of the pause in the annealing process does not improve the performance.

In Fig. 7 we compare the two versions of the quantum annealer, i.e., the D-Wave 2000Q and the D-Wave Advantage. We compared both versions in their best performing configuration, that is 20μs of annealing time and 10^4 runs, even though for some values of n other configurations might slightly outperform this one. For most instances, the performance of D-Wave Advantage is better than the one of D-Wave 2000Q, but the gap between the two is quite small. The anomalous case ($n = 7$) in which the average values of the metrics used seem to suggest that D-Wave 2000Q shows a better performance is related to numerical fluctuations. However, taking the experimental error into consideration, the conclusions drawn for the other cases are still valid. The few cases ($n = 7$) where D-Wave 2000Q shows better performances are explained by the experimental error, as shown by the error bars. However, this fact slightly contrasts our expectation of better performance of D-Wave Advantage suggested by its more compact minor embedding (less physical qubits used).

Both quantum annealers perform worse than SA. Moreover, SA in general performs better than any other classical algorithm tested (results in Appendix B). We can state the superior performances of classical hardware compared to the current quantum annealers, for the GED problem.

It is important to notice that Hybrid Annealer D-Wave Leap has performance close to the SA, even without ever outperforming it. The significance of this finding is

Table 3 Average resource usage for quantum annealers. Legend: *n* number of vertices, *L* logical qubits, *P* physical qubits, *CL* maximum chain length, *CS* chain strength

	D-Wave 2000				D-Wave Advantage			
n	L	P	CL	CS	L	P	CL	CS
3	9	22	3	1.44	9	11	2	1.44
4	16	65	5	1.76	16	33	3	1.76
5	25	175	9	2.03	25	77	4	2.03
6	36	377	14	2.27	36	159	6	2.27
7	49	711	21	2.48	49	296	9	2.48
8	64	1226	28	2.68	64	506	12	2.68
9	81	1474	28	2.84	81	832	16	2.87

Table 4 Average resource usage for variational algorithms. Legend: *n* number of vertices, *Q* number of qubits, *P* parameters, *D* depth, *S* size

		VQE						QAOA					
		$p=1$			$p=3$			$p=1$			$p=3$		
n	Q	P	D	S	P	D	S	P	D	S	P	D	S
3	9	36	17	54	72	37	144	2	40	96	6	88	258
4	16	64	31	152	128	65	424	2	79	272	6	119	792
5	25	100	49	350	200	101	1000	2	125	638	6	187	1876
6	36	144	71	702	288	145	2034	2	183	1220	6	275	3606
7	49	196	97	1274	392	197	3724	2	257	2109	6	385	6255
8	64	256	127	2144	512	257	6304	2	329	3625	6	494	11259
9	81	324	161	3402	648	325	10044	2	423	5535	6	635	16263

Table 5 Time To Solution for the two hardware configurations, measured in seconds. Legend: *n* number of vertices. A: 1μs per run × 10*k* runs, B: 20μs per run × 10*k* runs, C: 20μs per run × 1*k* runs, D: 500μs per run × 1*k* runs, E: 100μs per run × 1*k* runs with paused annealing

	D-Wave 2000Q					D-Wave Advantage				
n	A	B	C	D	E	A	B	C	D	E
3	0.01	0.20	0.02	0.50	0.10	0.01	0.20	0.02	0.50	0.10
4	0.01	0.20	0.02	0.50	0.10	0.01	0.20	0.02	0.50	0.10
5	0.01	0.23	0.02	0.57	0.12	0.01	0.23	0.02	0.57	0.11
6	0.03	0.25	0.04	0.72	0.14	0.01	0.23	0.04	0.57	0.18
7	0.05	0.80	0.08	1.61	0.32	0.03	0.80	0.11	2.00	0.32
8	0.17	0.45	0.33	1.14	0.32	0.02	0.45	0.17	8.33	1.67

weakened by the opaque internal working of D-Wave Leap, which might be perform SA itself.

Finally, we have identified which configuration promises the best tradeoff between quality of the solution and total annealing time. This observation is relevant because the cost of using a quantum annealer is proportional to the amount of time used by the machine.

We measure the *time to solution* (TTS) as

$$TTS = \frac{\text{\# of runs} \times \text{annealing time per run}}{\text{high-quality probability}}$$

for both D-Wave 2000 and D-Wave Advantage. It measures how much time in seconds is required on a quantum annealer to find a high-quality solution. In the literature you can find also different definitions of TTS (Ohzeki et al. 2019; Aramon et al. 2019), all representing the same concept. The results are shown in Table 5. Our experiments show that having the shortest annealing time with a large number of runs gives the best tradeoff in terms of TTS. For $n = 9$ vertices no experiment found any high-quality solution thus was not possible to estimate such probability.

6.3 Comparing variational algorithms

Figure 8 shows the comparison of the performance of VQE, QAOA, and SA. Considering the current state of NISQ devices, we have tested the variational approach on instances of GED with graphs having at most 5 vertices (thus our

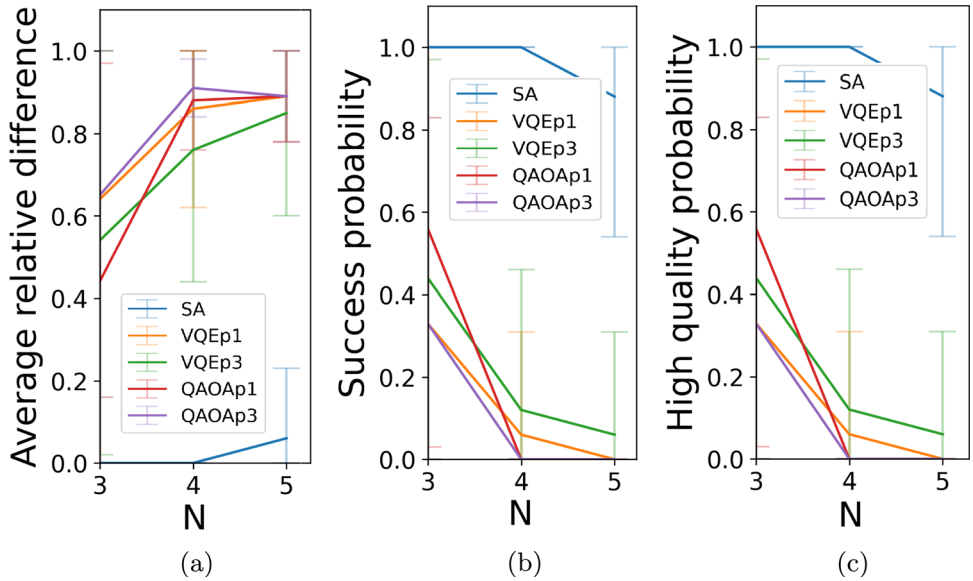

Fig. 8 Performance of variational algorithms to compute the GED for graphs with n vertices. In (**a**) we plot the mean relative difference, while in (**b**) we plot the success solution probability and in (**c**) the high-quality solution probability. The acronyms refer to the different algorithms as follows (the error bars represent the standard deviation): V1 is VQE with $p=1$, V3 is VQE with $p=3$, Q1 is QAOA with $p=1$, and Q3 is QAOA with $p=3$

circuit uses 25 qubits). For these small dimensions, the classical approach leads to much better solutions. In particular, no variational approaches are able to find an exact solution for instances having $n \geq 4$ vertices.

Increasing the number of repetitions may marginally improve the performances: VQE with $p=3$ outperforms VQE with $p=1$ for any choice of n, while for QAOA the configuration with $p=1$ performs equal or better than the one with $p=3$. Theoretically, in the limit of $p \to \infty$, the probability of success is guaranteed to be 1 (see Farhi et al. 2014). However, increasing the number of layers becomes really costly in terms of the evaluation of the expectation value of the cost Hamiltonian. In fact, the dimensions of the parameter search space scales, when no a priori symmetries are known, as $[-\pi, \pi]^p \times [-\pi, \pi]^p$. To gain insight into the low performances of the QAOA algorithm we consider the energy landscape of the associate Hamiltonian of the QUBO formulation of the GED of two graphs having four vertices and with $p=1$. For the case considered, the Hamiltonian H_C has a discrete spectrum with 88 eigenenergies, whose ground state is degenerate.

We report the energy landscape obtained in Fig. 9, where it is clear that the Hamiltonian does not have a unique global minima, leading to a low performance of the optimization technique provided by the QAOA for the QUBO formulation of the GED problem.

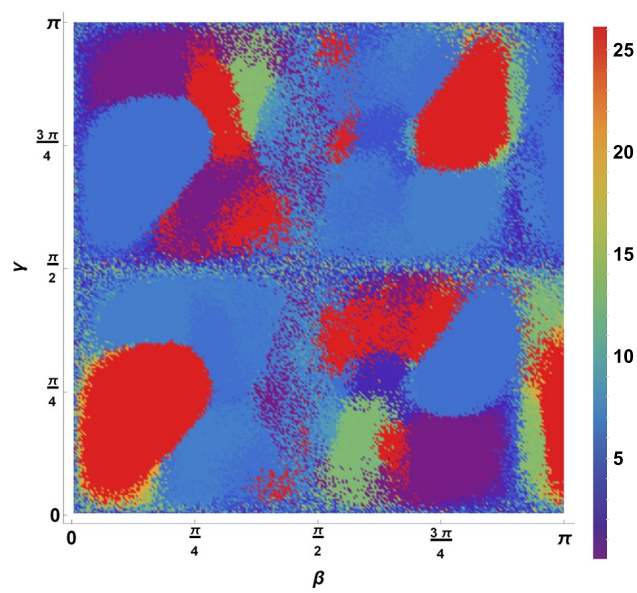

Fig. 9 Energy Landscape given by Eq. (16) for a Hamiltonian referring to a pair of graphs with $N=4$. Due to the symmetry of the system we have plotted the landscape only in the relevant range $[0, \pi] \times [0, \pi]$

7 Concluding remarks

Complex systems are nowadays ubiquitous in science. Very useful models for these systems are typically defined by representing them as graphs, i.e., collections of pairwise connected nodes. The nodes constitute the elementary "units" of the problem, while the edges take into account their interaction. Each edge can also be associated with a label

representing some quantitative or qualitative information. This versatility of the graph structures makes them a powerful tool in the most heterogeneous fields of research.

In this paper we have addressed the problem of quantitatively estimating the degree of similarity between a pair of graphs via the *Graph Edit Distance*. Computing GED is a task that requires the exploration of a space of solutions that is exponential in the size of the input graphs. It is therefore reasonable to consider approximate approaches to the problem, which are able to achieve acceptable approximations of the exact solution. We have thus investigated whether the resources offered by the quantum hardware currently available are plausible candidates to tackle this task. Our proof-of-principle analysis has had a twofold aim: on one side we have shown how to practically implement the computation of GED on both a quantum annealer and a gate-based quantum computer, and subsequently we have compared the results of running the same GED algorithm on the two types of hardware. Based on our results, the quantum annealer seems to be today a better platform for optimization problems written as QUBO problems.

We remark that, concerning the variational approach, we have used noiseless simulators to have a first insight into the feasibility of evaluating the GED with NISQ devices. Nowadays, the development of the gate-based quantum hardware as well as the specific software for optimization algorithms is still in its infancy. In this work we have reported some preliminary evidences that the main variational algorithms may not be suited for the type of problems as the one we have addressed in this paper. Overall, we are confronted with two contrasting aspects of the currently available resources. On one hand, classically simulating a quantum computer is expensive although it does not suffer from problems related to quantum physical resources, namely qubits and unitary gates acting for a sufficiently long time (Mandarino et al. 2018). On the other hand these quantum computational resources are very limited so as to make it impossible to exploit their full potential. However, we surmise that these detrimental effects will be overcome when quantum hardware with enough resources to allow error-correction schemes as a barrier against noise will be available. Another problem with variational algorithms is that it is necessary to guess the right encoding of the problem into the parameterized cost function that is evaluated using a quantum computer. This is a challenging but crucial task for the success of the solution scheme, together with the choice of the best classical optimizer for the parameter training phase. We surmise that a crucial step allowing for better performances of the variational quantum strategies will be achievable with algorithms approaching a higher order in the Suzuki-Trotter expansion combined with a low depth of the corresponding circuit.

In this direction, it was recently proposed a Quantum Natural Gradient Descent algorithm (Stokes et al. 2020; Gacon et al. 2021), which seems to offer enhanced performance by considering statistical information about the quantum circuit such as geometrical methods based on the Quantum Fisher Information (Abbas et al. 2021). Another seemingly promising strategy is to combine the two approaches that we have studied in this paper, namely using the quantum annealing to get preliminary results that can then be used to initialize a variational algorithm, as put forward for the case of the QAOA in Sack and Serbyn (2021). However, to the best of our knowledge, there are no available platforms allowing for a combined use of both types of quantum hardware.

We believe that the algorithms and the benchmarking of quantum annealers and gate-based quantum computers that we have presented in this paper can also be exploited for machine learning tasks. In fact, the algorithms we have devised in our implementations could be profitably incorporated in machine learning algorithms that deal with graph data, to obtain quantum algorithms that are more efficient than the standard classical machine learning strategies.

Appendix : A. Algorithm

In this Appendix we present the algorithms that we have implemented for creating our benchmark.

Algorithm 1 is used to construct the QUBO matrix that represents the QUBO formulation of the GED problem.

Algorithm 1 Construct QUBO matrix

```
Require: G, H graphs having the same number |n| of vertices
Ensure: Q n × n real matrix
  Q ← n × n symmetric matrix initialized to zero    ▷ Q[a, b] denotes both the
  a-th row, b-th col and a-th col, b-th row since the matrix is symmetric
  λ_h ← n^2 + 1
  λ_s ← 1
                        ▷ construct hard constraint (squared formula expanded
  for all i ∈ [0, n) do                                  ▷ row hard constraint
      for all j ∈ [0, n) do                                  ▷ x_{i,j}^2 − 2x_{i,j}
          ℓ ← i × n + j
          Q[ℓ, ℓ] ← Q[ℓ, ℓ] − λ_h
      for all j ∈ [0, n) do                                  ▷ 2x_{i,j}x_{i,j'}
          for all j' ∈ [j, n) do
              ℓ ← i × n + j
              ℓ' ← i × n + j'
              Q[ℓ, ℓ'] ← Q[ℓ, ℓ'] + 2λ_h
  for all j ∈ [0, n) do                                  ▷ column hard constraint
      for all i ∈ [0, n) do                                  ▷ x_{i,j}^2 − 2x_{i,j}
          ℓ ← i × n + j
          Q[ℓ, ℓ] ← Q[ℓ, ℓ] − λ_h
      for all i ∈ [0, n) do                                  ▷ 2x_{i,j}x_{i',j}
          for all i' ∈ [j, n) do
              ℓ ← i × n + j
              ℓ' ← i' × n + j
              Q[ℓ, ℓ'] ← Q[ℓ, ℓ'] + 2λ_h
  for all i, j ∈ E(G) do                                 ▷ P constraints
      for all i' ∈ [0, n) do
          for all j' ∈ [0, n) do
              if i', j' ∉ E(H) then
                  ℓ ← i × n + i'
                  ℓ' ← j × n + j'
                  Q[ℓ, ℓ'] ← Q[ℓ, ℓ'] + λ_s
  for all i', j' ∈ E(H) do                               ▷ Q constraints
      for all i ∈ [0, n) do
          for all j ∈ [0, n) do
              if i, j ∉ E(G) then
                  ℓ ← i × n + i'
                  ℓ' ← j × n + j'
                  Q[ℓ, ℓ'] ← Q[ℓ, ℓ'] + λ_s
  return Q
```

Algorithm 2 represents how the Simulating Annealing procedure we have used works.

Algorithm 2 Simulated annealing

Require: Q QUBO matrix sized $n \times n$
Require: $T_0 > 0$ initial temperature
Require: $\alpha \in (0, 1)$ temperature decrease factor
Require: $S > 0$ number of samples
Ensure: E optimal or sub-optimal cost for minimizing QUBO

```
samples ← empty list
energies ← empty list
repeat
    Generate x ∈ {0,1}^n randomly
    T ← T_0                                   ▷ initial temperature
    E ← x^T Q x                               ▷ initial energy
    repeat
        x' ← slightly perturbed x
        E' ← (x')^T Q (x')
        ΔE ← E' − E
        r ← random value between 0 and 1
        if ΔE ≤ 0 then                        ▷ accept the solution
            x ← x'
            E ← E'
        else if e^{−(ΔE)/T} > r then          ▷ accept the solution
            x ← x'
            E ← E'
        else
            do nothing                        ▷ reject solution
        T ← α · T                             ▷ decrease temperature
    until T ≈ 0
    append x to samples
    append E to energies
    S ← S − 1
until S = 0 ▷ On D-Wave's implementation you can define also a non-zero final temperature
return samples, energies                      ▷ alternatevely, return min(energies)
```

Algorithm 3 constructs the parametric quantum circuit needed to run the VQE algorithm.

Algorithm 3 Construct VQE circuit

Require: p repetition of the ansatz
Require: $\theta \in [-\pi, \pi]^{2pn}$ parameters vector
Ensure: Circuit for VQE

```
Create circuit of n qubits
for i ∈ 1, ..., p do
    for j ∈ 1, ..., n do
        Apply gate R_y(θ_{2in+2i}) on j-th qubit
        Apply gate R_y(θ_{2in+2i+1}) on j-th qubit
    for j ∈ 1, ..., n − 1 do
        for k ∈ i + 1, ..., n do
            Apply gate CX on j-th (control) and k-th (target) qubits
return circuit
```

Algorithm 4 constructs the parametric quantum circuit needed to run the QAOA algorithm.

Algorithm 4 Construct QAOA circuit

Require: $H_C = \sum_{i,j} J_{i,j}\sigma_i^z\sigma_j^z$ defined on n variables
Require: p number of repetition of the Hamiltonians
Require: $\gamma \in [-\pi, \pi]^p$ parameter vector
Require: $\beta \in [-\pi, \pi]^p$ parameter vector
Ensure: Circuit for QAOA

```
Create circuit of n qubits                          ▷ Circuit starts in state |0⟩^⊗n
for i ∈ 1, ..., n do                                ▷ Circuit evolves in state |+⟩^⊗n
    Apply gate H on i-th qubit
for i ∈ 1, ..., p do
    for J_{i,j} σ_i^z σ_j^z in H_c do               ▷ Construct U_C(γ_i) = e^{-iγ_i H_C}
        if i = j then
            Apply gate R_Z(γ_i J_{i,j}) on i, j-th qubits
        else
            Apply gate R_ZZ(γ_i J_{i,j}) on i, j-th qubits
    for i ∈ 1, ..., n do                            ▷ Construct U_B(γ_i) = e^{-iβ_i H_B}
        Apply gate R_X(β_i) on i-th qubit
    return circuit
```

Appendix : B. Detailed results

Figures 10, 11 and 12 compare the performances of classical heuristics. All these approaches are detailed explained in Blumenthal and Gamper (2020).

Figure 10 contains the Average relative difference, as defined in Section 5 by Eq. (14).

Figure 11 contains the Average success probability, as defined in Section 6, which is the percentage of runs having null relative difference over all runs.

Figure 12 contains the Average high-quality probability, as defined in Section 6, which is the percentage of experiments having results with relative difference ≤ 0.2.

Figures 13, 14 and 15 compare the performances of simulated annealing, quantum annealing, variational algorithms.

Figure 13 contains the Average relative difference, as defined in Section 5 by Eq. (14).

Figure 14 contains the Average success probability, as defined in Section 6, which is the percentage of runs having null relative difference over all runs.

V	B1	B2	B3	B4	B5	P1	H1	R1	A1	W1	I1	B6	S1	N1	R2	B7	R3	B8	SA	H2	S2
3	0.00	0.00	0.00	0.00	1.00	1.00	1.00	0.00	0.11	0.00	0.22	0.00	0.00	0.22	0.22	0.32	0.00	0.00	0.00	1.00	0.00
4	0.08	0.08	0.00	0.08	1.00	1.00	1.00	0.08	0.04	0.08	0.10	0.08	0.16	0.16	0.29	0.19	0.06	0.00	0.00	1.00	0.08
5	0.27	0.27	0.22	0.27	1.00	1.00	1.00	0.16	0.22	0.27	0.28	0.27	0.28	0.29	0.26	0.34	0.00	0.17	0.00	1.00	0.27
6	0.29	0.29	0.24	0.29	1.00	1.00	1.00	0.17	0.16	0.26	0.19	0.29	0.40	0.34	0.50	0.49	0.15	0.28	0.00	1.00	0.29
7	0.34	0.34	0.22	0.34	1.00	1.00	1.00	0.19	0.36	0.34	0.27	0.34	0.36	0.34	0.60	0.42	0.29	0.31	0.00	1.00	0.35
8	0.19	0.19	0.19	0.19	1.00	1.00	1.00	0.17	0.18	0.19	0.35	0.19	0.43	0.21	0.32	0.46	0.21	0.23	0.01	1.00	0.19
9	0.22	0.22	0.17	0.22	1.00	1.00	1.00	0.17	0.30	0.21	0.18	0.22	0.40	0.24	0.32	0.54	0.25	0.34	0.07	1.00	0.24

Fig. 10 Average relative difference (lower is better), Legend: V: number of vertices, B1: Branch, B2: Branch Fast, B3: Branch Tight, B4: Branch Uniform, B5: Branch Compact, P1: Partition, H1: Hybrid, R1: Ring, A1: ANCHOR_AWARE_GED, W1: Walks, I1: IPFP, B6: BIPARTITE, S1: SUBGRAPH, N1: NODE, R2: RING_ML, B7: BIPARTITE_ML, R3: REFINE, B8: BP_BEAM, SA: Simulated Annealing (GEDLib implementation, not D-Wave's) H2: HED, S2: STAR

V	B1	B2	B3	B4	B5	P1	H1	R1	A1	W1	I1	B6	S1	N1	R2	B7	R3	B8	SA	H2	S2
3	1.00	1.00	1.00	1.00	0.00	0.00	0.00	1.00	0.89	1.00	0.78	1.00	1.00	0.78	0.78	0.67	1.00	1.00	1.00	0.00	1.00
4	0.88	0.88	1.00	0.88	0.00	0.00	0.00	0.88	0.94	0.88	0.88	0.88	0.75	0.75	0.62	0.75	0.94	1.00	1.00	0.00	0.88
5	0.62	0.62	0.69	0.62	0.00	0.00	0.00	0.75	0.69	0.62	0.62	0.62	0.62	0.62	0.62	0.56	1.00	0.81	1.00	0.00	0.62
6	0.44	0.44	0.56	0.44	0.00	0.00	0.00	0.56	0.75	0.44	0.81	0.44	0.31	0.38	0.25	0.25	0.81	0.69	1.00	0.00	0.38
7	0.38	0.38	0.50	0.38	0.00	0.00	0.00	0.56	0.50	0.31	0.62	0.38	0.44	0.38	0.12	0.19	0.56	0.56	1.00	0.00	0.31
8	0.38	0.38	0.38	0.38	0.00	0.00	0.00	0.56	0.75	0.44	0.56	0.38	0.19	0.38	0.31	0.06	0.75	0.75	0.94	0.00	0.44
9	0.38	0.38	0.38	0.38	0.00	0.00	0.00	0.50	0.62	0.25	0.75	0.38	0.19	0.38	0.25	0.00	0.62	0.50	0.69	0.00	0.25

Fig. 11 Average Success probability (higher is better), Legend: V: number of vertices, B1: Branch, B2: Branch Fast, B3: Branch Tight, B4: Branch Uniform, B5: Branch Compact, P1: Partition, H1: Hybrid, R1: Ring, A1: ANCHOR_AWARE_GED, W1: Walks, I1: IPFP, B6: BIPARTITE, S1: SUBGRAPH, N1: NODE, R2: RING_ML, B7: BIPARTITE_ML, R3: REFINE, B8: BP_BEAM, SA: Simulated Annealing (GEDLib implementation, not D-Wave's) H2: HED, S2: STAR

V	B1	B2	B3	B4	B5	P1	H1	R1	A1	W1	I1	B6	S1	N1	R2	B7	R3	B8	SA	H2	S2
3	1.00	1.00	1.00	1.00	0.00	0.00	0.00	1.00	0.89	1.00	0.78	1.00	1.00	0.78	0.78	0.67	1.00	1.00	1.00	0.00	1.00
4	0.88	0.88	1.00	0.88	0.00	0.00	0.00	0.88	0.94	0.88	0.88	0.88	0.75	0.75	0.62	0.75	0.94	1.00	1.00	0.00	0.88
5	0.62	0.62	0.69	0.62	0.00	0.00	0.00	0.75	0.69	0.62	0.62	0.62	0.62	0.62	0.62	0.56	1.00	0.81	1.00	0.00	0.62
6	0.44	0.44	0.56	0.44	0.00	0.00	0.00	0.62	0.75	0.44	0.81	0.44	0.31	0.38	0.25	0.25	0.81	0.69	1.00	0.00	0.38
7	0.38	0.38	0.50	0.38	0.00	0.00	0.00	0.62	0.50	0.31	0.62	0.38	0.44	0.38	0.25	0.38	0.56	0.56	1.00	0.00	0.31
8	0.62	0.62	0.62	0.62	0.00	0.00	0.00	0.69	0.75	0.62	0.56	0.62	0.44	0.62	0.62	0.44	0.75	0.75	1.00	0.00	0.56
9	0.75	0.75	0.75	0.75	0.00	0.00	0.00	0.69	0.62	0.69	0.75	0.75	0.38	0.62	0.56	0.19	0.62	0.56	0.88	0.00	0.62

Fig. 12 Average high-quality (<20%) probability (higher is better), Legend: V: number of vertices, B1: Branch, B2: Branch Fast, B3: Branch Tight, B4: Branch Uniform, B5: Branch Compact, P1: Partition, H1: Hybrid, R1: Ring, A1: ANCHOR_AWARE_GED, W1: Walks, I1: IPFP, B6: BIPARTITE, S1: SUBGRAPH, N1: NODE, R2: RING_ML, B7: BIPARTITE_ML, R3: REFINE, B8: BP_BEAM, SA: Simulated Annealing (GEDLib implementation, not D-Wave's) H2: HED, S2: STAR

		D-Wave 2000					D-Wave Advantage					D-Wave	VQE		QAOA	
V	SA	D1	D20	D21	D500	D100	D1	D20	D21	D500	D100	Leap	$p=1$	$p=3$	$p=1$	$p=3$
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.64	0.54	0.44	0.65
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.86	0.76	0.88	0.91
5	0.06	0.20	0.06	0.06	0.06	0.09	0.06	0.06	0.06	0.06	0.06	0.06	0.89	0.85	0.89	0.89
6	0.00	0.43	0.14	0.35	0.24	0.23	0.16	0.10	0.29	0.11	0.25	0.00	-	-	-	-
7	0.05	0.60	0.48	0.50	0.43	0.47	0.49	0.51	0.61	0.51	0.45	0.05	-	-	-	-
8	0.00	0.80	0.52	0.80	0.44	0.53	0.47	0.52	0.77	0.68	0.81	0.06	-	-	-	-
9	0.10	0.87	0.84	1.00	1.00	1.00	0.87	0.79	0.87	0.81	0.87	0.13	-	-	-	-

Fig. 13 Average relative difference (lower is better). Legend: V: number of vertices, SA: Simulated Annealing (D-Wave's implementation), D1: annealing 1μs × 10^4 runs, D20: annealing 20μs × 10^4 runs, D21: annealing 20μs × 10^3 runs, D500: annealing 500μs × 10^3 runs, D100: annealing 100μs paused in the middle, × 10^3 runs, p: number of repetition of the variational form

		D-Wave 2000					D-Wave Advantage					D-Wave	VQE		QAOA	
V	SA	D1	D20	D21	D500	D100	D1	D20	D21	D500	D100	Leap	$p = 1$	$p = 3$	$p = 1$	$p = 3$
3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.33	0.44	0.56	0.33
4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.06	0.12	0.00	0.00
5	0.88	0.69	0.88	0.88	0.88	0.81	0.88	0.88	0.88	0.88	0.88	0.88	0.00	0.06	0.00	0.00
6	1.00	0.38	0.69	0.38	0.69	0.56	0.62	0.81	0.56	0.75	0.56	1.00	-	-	-	-
7	0.88	0.00	0.12	0.12	0.25	0.19	0.19	0.06	0.06	0.19	0.12	0.88	-	-	-	-
8	1.00	0.06	0.19	0.00	0.25	0.19	0.12	0.19	0.00	0.06	0.00	0.88	-	-	-	-
9	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.69	-	-	-	-

Fig. 14 Average success probability (higher is better). Legend: V: number of vertices, SA: Simulated Annealing (D-Wave's implementation), D1: annealing 1μs × 10^4 runs, D20: annealing 20μs × 10^4 runs, D21: annealing 20μs × 10^3 runs, D500: annealing 500μs × 10^3 runs, D100: annealing 100μs paused in the middle, × 10^3 runs, p: number of repetition of the variational form

		D-Wave 2000					D-Wave Advantage					D-Wave	VQE		QAOA	
V	SA	D1	D20	D21	D500	D100	D1	D20	D21	D500	D100	Leap	$p = 1$	$p = 3$	$p = 1$	$p = 3$
3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.33	0.44	0.56	0.33
4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.06	0.12	0.00	0.00
5	0.88	0.69	0.88	0.88	0.88	0.81	0.88	0.88	0.88	0.88	0.88	0.88	0.00	0.06	0.00	0.00
6	1.00	0.38	0.81	0.50	0.69	0.69	0.75	0.88	0.56	0.88	0.56	1.00	-	-	-	-
7	0.88	0.19	0.25	0.25	0.31	0.31	0.31	0.25	0.19	0.25	0.31	0.88	-	-	-	-
8	1.00	0.06	0.44	0.06	0.44	0.31	0.44	0.44	0.12	0.06	0.06	0.88	-	-	-	-
9	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.00	0.81	-	-	-	-

Fig. 15 Average high-quality probability (higher is better). Legend: V: number of vertices, SA: Simulated Annealing (D-Wave's implementation), D1: annealing 1μs × 10^4 runs, D20: annealing 20μs × 10^4 runs, D21: annealing 20μs × 10^3 runs, D500: annealing 500μs × 10^3 runs, D100: annealing 100μs paused in the middle, × 10^3 runs, p: number of repetition of the variational form

Figure 15 contains the Average high-quality probability, as defined in Section 6, which is the percentage of experiments having results with relative difference ≤ 0.2.

Acknowledgements We would like to thank CINECA for making available to us the necessary resources for running our tests (QPU D-Wave and classical supercomputers for simulations), through the Italian SuperComputing Resource Allocation - ISCRA - programme. A.M. acknowledges the University of Verona for financial support throughout Mobility Grant No. INTAT20DIPIERRO, the Foundation for Polish Science (FNP) for the IRAP project ICTQT (Contract No. 2018/MAB/5), cofinanced by the EU Smart Growth Operational Programme, and the Polish National Science Center for the grant MINIATURA DEC-2020/04/X/ST2/01794.

Declarations

Conflict of interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Abbas A, Sutter D, Zoufal C, Lucchi A, Figalli A, Woerner S (2021) The power of quantum neural networks. Nature Computational Science 1(6):403–409

Abu-Aisheh Z, Raveaux R, Ramel J. -Y., Martineau P (2015) An exact graph edit distance algorithm for solving pattern recognition problems. In: Proceedings of the International Conference on Pattern Recognition Applications and Methods - Volume 1. ICPRAM 2015, 271–278, SCITEPRESS - Science and Technology Publications, Lda. https://doi.org/10.5220/0005209202710278

Albash T, Lidar DA (2018) Adiabatic quantum computation. Rev Mod Phys 90:015002. https://doi.org/10.1103/RevModPhys.90.015002

Alexeev Y, Bacon D, Brown KR, Calderbank R, Carr LD, Chong FT, DeMarco B, Englund D, Farhi E, Fefferman B, Gorshkov AV, Houck A, Kim J, Kimmel S, Lange M, Lloyd S, Lukin MD, Maslov D, Maunz P, Monroe C, Preskill J, Roetteler M, Savage MJ, Thompson J (2021) Quantum computer systems for scientific discovery. PRX Quantum 2:017001. https://doi.org/10.1103/PRXQuantum.2.017001

Almohamad HA, Duffuaa SO (1993) A linear programming approach for the weighted graph matching problem. IEEE Trans Pattern Anal Mach Intell 15(5):522–525. https://doi.org/10.1109/34.211474

Apolloni B, Carvalho C, De Falco D (1989) Quantum stochastic optimization. Stoch Process Appl 33(2):233–244

Aramon M, Rosenberg G, Valiante E, Miyazawa T, Tamura H, Katzgraber HG (2019) Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. Frontiers in Physics 7:48

Asproni L, Caputo D, Silva B, Fazzi G, Magagnini M (2020) Accuracy and minor embedding in subqubo decomposition with fully connected large problems: a case study about the number partitioning problem. Quantum Machine Intell 2(1):1–7

Babai L (2016) Graph isomorphism in quasipolynomial time. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing. STOC '16. https://doi.org/10.1145/2897518.2897542. Association for Computing Machinery, pp 684–697

Batagelj V, Brandes U (2005) Efficient generation of large random networks. Phys Rev E Stat Nonlinear Soft Matter Phys 71:036113. https://doi.org/10.1103/PhysRevE.71.036113

Bellet A, Sebban M, Habrard A (2011) An experimental study on learning with good edit similarity functions. In: 2011 IEEE 23rd international conference on tools with artificial intelligence. IEEE, pp 126–133

Bengtsson A, Vikstål P, Warren C, Svensson M, Gu X, Kockum AF, Krantz P, Križan C, Shiri D, Svensson I-M, Tancredi G, Johansson G, Delsing P, Ferrini G, Bylander J (2020) Improved success probability with greater circuit depth for the quantum approximate optimization algorithm. Phys Rev Appl 14:034010. https://doi.org/10.1103/PhysRevApplied.14.034010

Blumenthal DB, Boria N, Gamper J, Bougleux S, Brun L (2020) Comparing heuristics for graph edit distance computation. The VLDB Journal 29(1):419–458

Blumenthal DB, Bougleux S, Gamper J, Brun L (2019) Gedlib: a c++ library for graph edit distance computation. In: Conte D, Ramel J-Y, Foggia P (eds) Graph-based representations in pattern recognition. Springer, pp 14–24

Blumenthal DB, Gamper J (2020) On the exact computation of the graph edit distance. Pattern Recogn Lett 134:46–57

Bougleux S, Gaüzère B, Brun L (2017) A hungarian algorithm for error-correcting graph matching. In: International workshop on graph-based representations in pattern recognition. Springer, pp 118–127

Calude CS, Dinneen MJ, Hua R (2017) Qubo formulations for the graph isomorphism problem and related problems. Theor Comput Sci 701(C):54–69

Chen H, Lidar DA (2020) Why and when pausing is beneficial in quantum annealing. Physical Review Applied 14(1):014100

Chiesa A, Tacchino F, Grossi M, Santini P, Tavernelli I, Gerace D, Carretta S (2019) Quantum hardware simulating four-dimensional inelastic neutron scattering. Nat Phys 15(5):455–459

Childs AM, Schoute E, Unsal CM (2019) Circuit transformations for quantum architectures. arXiv:1902.09102

Choi V (2008) Minor-embedding in adiabatic quantum computation: i. the parameter setting problem. Quantum Inf Process 7(5):193–209

Conte D, Foggia P, Sansone C, Vento M (2003) Graph matching applications in pattern recognition and image processing. In: Proceedings 2003 international conference on image processing (Cat. No.03CH37429). https://doi.org/10.1109/ICIP.2003.1246606, vol 2, p 21

Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. Int J Pattern Recognit Artif Intell 18(03):265–298

Crippa L, Tacchino F, Chizzini M, Aita A, Grossi M, Chiesa A, Santini P, Tavernelli I, Carretta S (2021) Simulating static and dynamic properties of magnetic molecules with prototype quantum computers. Magnetochemistry 7(8):117

D-Wave Systems Inc (2020) Qpu-specific physical properties: Dw_2000q_6. Technical report, D-Wave Systems Inc. https://docs.dwavesys.com/docs/latest/_downloads/92a6dea183a90d7d4f6d305c94812c8d/09-1215A-B_QPU_Properties_DW_2000Q_6.pdf

D-Wave Systems Inc (2020) Qpu-specific physical properties: Advantage_system1.1. Technical report, D-Wave Systems Inc. https://docs.dwavesys.com/docs/latest/_downloads/9a1b594d84370df6c0a09d00a5b72661/09-1237A-A_QPU_Properties_Advantage_system1_1.pdf

Farhi E, Goldstone J, Gutmann S (2014). arXiv:1411.4028

Filatov A, Gitis A, Kil I (1995) Graph-based handwritten digit string recognition. In: Proceedings of 3rd international conference on document analysis and recognition. https://doi.org/10.1109/ICDAR.1995.602033, vol 2, pp 845–8482

Fortin S (1996) The graph isomorphism problem. Technical report, University of Alberta. https://web.archive.org/web/20130616212259/http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.9419&rep=rep1&type=pdf

Funcke L, Hartung T, Jansen K, Kühn S., Stornati P (2021) Dimensional expressivity analysis of parametric quantum circuits. Quantum 5:422

Fyrbiak M, Wallat S, Reinhard S, Bissantz N, Paar C (2020) Graph similarity and its applications to hardware security. IEEE Trans Comput 69(4):505–519. https://doi.org/10.1109/TC.2019.2953752

Gacon J, Zoufal C, Carleo G, Woerner S (2021) Simultaneous perturbation stochastic approximation of the quantum fisher information. Quantum 5

Gao X, Xiao B, Tao D, Li X (2010) A survey of graph edit distance. Pattern Anal Appl 13:113–129. https://doi.org/10.1007/s10044-008-0141-y

Hagberg AA, Schult DA, Swart PJ (2008) Exploring network structure, dynamics, and function using networkx. In: Varoquaux G, Vaught T, Millman J (eds) Proceedings of the 7th python in science conference, Pasadena, CA USA, pp 11–15

Hauke P, Katzgraber HG, Lechner W, Nishimori H, Oliver WD (2020) Perspectives of quantum annealing: methods and implementations. Rep Prog Phys 83(5):054401. https://doi.org/10.1088/1361-6633/ab85b8

Hoos HH, Tsang E (2006) Chapter 5 - local search methods. In: Rossi F, van Beek P, Walsh T (eds) Handbook of constraint programming. foundations of artificial intelligence. https://doi.org/10.1016/S1574-6526(06)80009-X. https://www.sciencedirect.com/science/article/pii/S157465260680009X, vol 2. Elsevier, pp 135–167

Huembeli P, Arrazola JM, Killoran N, Mohseni M, Wittek P (2022) The physics of energy-based models. Quantum Machine Intell 4(1):1–13

IBM (2021) Qiskit: An open-source framework for quantum computing. https://doi.org/10.5281/zenodo.2573505

Justice D, Hero A (2006) A binary linear programming formulation of the graph edit distance. IEEE Trans Pattern Anal Mach Intell 28(8):1200–1214. https://doi.org/10.1109/TPAMI.2006.152

Kadowaki T, Nishimori H (1998) Quantum annealing in the transverse ising model. Phys Rev E 58:5355–5363. https://doi.org/10.1103/PhysRevE.58.5355

Kandala A, Mezzacapo A, Temme K, Takita M, Brink M, Chow JM, Gambetta JM (2017) Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature 549(7671):242–246

Kato T (1950) On the adiabatic theorem of quantum mechanics. J Phys Soc Japan 5(6):435–439. https://doi.org/10.1143/JPSJ.5.435

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Kisku DR, Gupta P, Sing JK (2010) Face recognition using sift descriptor under multiple paradigms of graph similarity constraints. Int J Multimed Ubiquitous Eng 5(4):1–18

Kochenberger G, Hao J. -K., Glover F, Lewis M, Lü Z., Wang H, Wang Y (2014) The unconstrained binary quadratic programming problem: a survey. J Comb Optim 28(1):58–81

Lucas A (2014) Ising formulations of many np problems. Frontiers in Physics 2:5. https://doi.org/10.3389/fphy.2014.00005

Mandarino A, Linowski T, Życzkowski K. (2018) Bipartite unitary gates and billiard dynamics in the weyl chamber. Phys Rev A 98:012335. https://doi.org/10.1103/PhysRevA.98.012335

Marshall J, Venturelli D, Hen I, Rieffel EG (2019) Power of pausing: Advancing understanding of thermalization in experimental quantum annealers. Physical Review Applied 11(4). https://doi.org/10.1103/physrevapplied.11.044083

Martina S, Buffoni L, Gherardini S, Caruso F (2022) Learning the noise fingerprint of quantum devices. Quantum Machine Intell 4(1):1–12

Mauerer W, Scherzinger S (2022) 1-2-3 reproducibility for quantum software experiments

Mbeng GB, Fazio R, Santoro G (2019) Quantum Annealing: a journey through Digitalization, Control and hybrid Quantum Variational schemes

McClean JR, Romero J, Babbush R, Aspuru-Guzik A (2016) The theory of variational hybrid quantum-classical algorithms. New J Phys 18(2):023023

McGeoch CC (2014) Adiabatic quantum computation and quantum annealing: Theory and practice. Synthesis Lectures on Quantum Computing 5(2):1–93

McGeoch C, Farré P (2020) The d-wave advantage system: An overview. Technical report, D-Wave Systems Inc. https://www.dwavesys.com/media/s3qbjp3s/14-1049a-a_thed-wave_advantage_system_an_overview.pdf

Mengoni R, Di Pierro A (2019) Kernel methods in quantum machine learning. Quantum Machine Intelligence 1(3):65–71

Mengoni R, Incudini M, Di Pierro A (2021) Facial expression recognition on a quantum computer. Quant Machine Intell 3(1):1–11

Mézard M, Parisi G, Virasoro MA (1987) Spin glass theory and beyond: an introduction to the replica method and its applications vol. 9 World Scientific Publishing Company

Mheich A, Hassan M, Gripon V, Khalil M, Berrou C, Dufor O, Wendling F (2015) A novel algorithm for measuring graph similarity: application to brain networks. In: 7th Annual International IEEE EMBS conference on neural engineering (NER). International IEEE EMBS conference on neural engineering. https://hal-univ-rennes1.archives-ouvertes.fr/hal-01380111. IEEE, pp 1068–1071

Neuhaus M, Bunke H (2004) An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification. In: Fred A, Caelli TM, Duin RPW, Campilho AC, De Ridder D (eds) Structural syntactic and statistical pattern recognition. Springer, pp 180–189

Neuhaus M, Bunke H (2007) Bridging the gap between graph edit distance and kernel machines. World scientific publishing co. Inc.

Ohzeki M, Miki A, Miyama MJ, Terabe M (2019) Control of automated guided vehicles without collision by quantum annealer and digital devices. Frontiers in Computer Science 1:9

Pardalos PM, Jha S (1992) Complexity of uniqueness and local search in quadratic 0–1 programming. Oper Res Lett 11(2):119–123. https://doi.org/10.1016/0167-6377(92)90043-3

Passarelli G, Cataudella V, Lucignano P (2019) Improving quantum annealing of the ferromagnetic p-spin model through pausing. Phys Rev B 100(2):024302

Pellow-Jarman A, Sinayskiy I, Pillay A, Petruccione F (2021) A comparison of various classical optimizers for a variational quantum linear solver. Quantum Inf Process 20(6):1–14

Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love PJ, Aspuru-Guzik A, O'brien JL (2014) A variational eigenvalue solver on a photonic quantum processor. Nature communications 5(1):1–7

Powell MJ (1994) A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in optimization and numerical analysis. Springer, pp 51–67

Preskill J (2018) Quantum Computing in the NISQ era and beyond. Quantum 2:79. https://doi.org/10.22331/q-2018-08-06-79

Sack SH, Serbyn M (2021) Quantum annealing initialization of the quantum approximate optimization algorithm. Quantum 5:491. https://doi.org/10.22331/q-2021-07-01-491

Sanfeliu A, Fu K-S (1983) A distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst Man Cybern SMC 13(3):353–362. https://doi.org/10.1109/TSMC.1983.6313167

Silva C, Aguiar A, Lima P, Dutra I (2020) Mapping graph coloring to quantum annealing. Quantum Machine Intell 2(2):1–19

Stokes J, Izaac J, Killoran N, Carleo G (2020) Quantum natural gradient. Quantum 4:269

Vikstål P, Grönkvist M, Svensson M, Andersson M, Johansson G, Ferrini G (2020) Applying the quantum approximate optimization algorithm to the tail-assignment problem. Phys Rev Applied 14:034009. https://doi.org/10.1103/PhysRevApplied.14.034009

Vogelstein JT, Conroy JM, Lyzinski V, Podrazik LJ, Kratzer SG, Harley ET, Fishkind DE, Vogelstein RJ, Priebe CE (2015) Fast approximate quadratic programming for graph matching. PLOS ONE 10:1–17. https://doi.org/10.1371/journal.pone.0121002

Willsch M, Willsch D, Jin F, De Raedt H, Michielsen K (2020) Benchmarking the quantum approximate optimization algorithm. Quantum Inf Process 19(7):1–24

Wilson M, Stromswold R, Wudarski F, Hadfield S, Tubman NM, Rieffel EG (2021) Optimizing quantum heuristics with meta-learning. Quantum Machine Intell 3(1):1–14

Wiskott L, Krüger N, Kuiger N, Von Der Malsburg C (1997) Face recognition by elastic bunch graph matching. IEEE Trans Pattern Anal Machine Intell 19(7):775–779

Zaslavskiy M (2010) Graph matching and its application in computer vision and bioinformatics. PhD thesis, École Nationale Supérieure des Mines de Paris. Thèse de doctorat dirigée par Bach, Francis et Vert, Jean-Philippe Bioinformatique Paris, ENMP 2010. https://pastel.archives-ouvertes.fr/pastel-00006121/

Zeng Z, Tung AK, Wang J, Feng J, Zhou L (2009) Comparing stars: on approximating graph edit distance. Proceedings of the VLDB Endowment 2(1):25–36

Zhou L, Wang S-T, Choi S, Pichler H, Lukin MD (2020) Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. Physical Review X 10 (2):021067

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.