QBoost for regression problems: solving partial differential equations

Caio B. D. Góes · Thiago O. Maciel · Giovani G. Pollachini · Juan P. L. C. Salazar · Rafael G. Cuenca · Eduardo I. Duzzioni

Received: 10 October 2022 / Accepted: 30 January 2023 / Published online: 25 February 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
A hybrid algorithm based on machine learning and quantum ensemble learning is proposed to find an approximate solution to a partial differential equation with good precision and favorable scaling in the required number of qubits. The classical component consists in training several regressors (weak-learners), capable of solving a partial differential equation approximately using machine learning. The quantum component consists in adapting the QBoost algorithm to solve regression problems to build an ensemble of classical learners. We have successfully applied our framework to solve the 1D Burgers’ equation with viscosity, showing that the quantum ensemble method...
really improves the solutions produced by classical weak-learners. We also implemented the algorithm on the D-Wave Systems, confirming the good performance of the quantum solution compared to the simulated annealing and exact solver methods.

**Keywords** Quantum computing · Partial differential equations · QBoost · Neural network

## 1 Introduction

Classical machine learning (ML) techniques have recently become an important tool for addressing problems in quantum mechanics and in the physical science [1], with applications in solving the Schrödinger’s equation [2], quantum state tomography [3, 4], quantum control [5], quantum phase transitions [6, 7], quantum chemistry [8], astronomical object recognition [9], and validation of quantum experiments [10]. In a nutshell, ML is able to solve difficult problems using complex models [11] which otherwise are hard for human mind to conceive.

As quantum mechanics can benefit from classical ML techniques, ML can also be improved by quantum mechanics. We find several instances of research that translate the various classical ML models for quantum computing with the aim of obtaining some speedup in training [12, 13] and data storage [14–16]. Some examples are quantum K-Nearest-Neighbor [17, 18], quantum decision trees [19, 20], quantum generative adversarial network [21, 22], and quantum kernel methods [23, 24].

The main idea of the ML model based on ensemble learning is to add several different models using weights to create a combined model that is better than all of its constituents individually [25, 26]. Ensemble learning models have been used for problems of classification [27] and regression [28] alike. The QBoost [29–31] algorithm was the pioneer in translating the ensemble learning model to the quantum realm by addressing classification problems. Subsequently, more ensemble learning models have been quantized [32, 33]. This technique becomes attractive when we have access to some particular solutions of the problem, as, for instance, we know analytical, numerical, or experimental solutions of a given problem, and then we find solutions that were not in the training set.

In the field of predictions by regression, we can find several applications of ML algorithms for “real-world” problems ranging from weather forecast [34, 35], traffic flow [36, 37], solar radiation estimation [38, 39] to aerodynamic applications [40–44]. In general, fluid mechanics problems rely on the solution of partial differential equations (PDEs). There are several classical ML methods in the literature that propose to find the solution of these PDEs [45–49].

In this work, we propose an adaptation of the QBoost algorithm for regression problems and apply the algorithm to solve the 1D viscous Burgers’ equation. The 1D Burgers’ viscous equation is often used as a proxy for the Navier–Stokes equation, as they both share nonlinearity in the convective term and a diffusive term. It is also used to assess the accuracy of numerical schemes, owing to the existence of analytical solutions. Our goal is to explore the feasibility of a QBoost algorithm running on quantum hardware to find optimal weights to the weak-learner ensemble model described
above. This problem is motivated by the growing interest in to use quantum hardware to solve real-world applications. In this particular case, motivation is provided by the recent Airbus Quantum Computing Challenge.¹

The paper is organized as follows: in 2 we describe the adaptation of the QBoost algorithm to deal with regression problems; in 3 we explain the hybrid classical-quantum algorithm used to find the solution of the PDE; in 4 we present and discuss the results obtained. The conclusions and future perspectives are given in 5.

2 QBoost for regression problems

Quantum annealing is an optimization procedure that exploits the phenomenon of quantum fluctuations and quantum tunneling to find the minimum value of an objective function probabilistically, since the system is at a nonzero temperature [50, 51]. The basic principle of quantum annealing is grounded on the adiabatic theorem, which tells us that if the system starts at the ground state of a Hamiltonian, $H_I$, that is known and easy to prepare and then it is allowed to evolve adiabatically to a final Hamiltonian, $H_F$, it would remain in its eigenstate [50], containing the solution of the desired problem. In other words, the Hamiltonian $H(t)$ changes in time according to,

$$H(t) = A(t)H_I + B(t)H_F,$$

where $A(t)$ and $B(t)$ define the annealing schedule and must satisfy the following restrictions: $A(0) \neq 0$, $B(0) = 0$ and $B(T) \neq 0$, $A(T) = 0$, where $T$ is the total evolution time [50, 51]. As the objective function to be minimized is encoded in $H_F$, after the annealing process we have the minimum of this function.

The adiabatic theorem [52, 53], which has been stated in many different ways (see [50] for a review), tells us that for the occurrence of a transitionless state evolution, the total evolution time $T$ must satisfy [54] ($\hbar = 1$)

$$T \gg \max_{0 \leq s \leq 1} \left| \frac{\langle k(s) | \frac{dH(s)}{ds} | m(s) \rangle}{\left( \min_{0 \leq s \leq 1} |\Delta_{mk}(s)| \right)^2} \right|,$$

where $\Delta_{mk} = E_m - E_k$, $E_m$ and $E_k$ are the eigenenergies associated with the eigenstates $|m(s)\rangle$ and $|k(s)\rangle$, respectively, $s = t/T$ is a dimensionless time, and $t$ is the current time.

The Ising Hamiltonian in a transverse field is typically used to perform the quantum annealing process [51], as is the case of D-Wave quantum process units (QPU) [55],

¹ See https://www.airbus.com/en/innovation/disruptive-concepts/quantum-technologies/airbus-quantum-computing-challenge.
given by,

\[ H(t) = -\frac{A(t)}{2} \sum_i \sigma_i^x + \frac{B(t)}{2} \left( \sum_i d_i \sigma_i^z + \sum_{i>j} D_{ij} \sigma_i^z \sigma_j^z \right), \]

(3)

where \( \sigma_i^x \) and \( \sigma_i^z \) are Pauli matrices acting on the \( i \)-th qubit in \( x \) and \( z \) directions, \( d_i \) is the transverse field applied to the \( i \)-th qubit, and \( D_{ij} \) is the coupling constant between the \( i \)-th and \( j \)-th qubits.

In this work, we propose a variant of the QBoost algorithm \([29–31]\) adequate for regression problems. In QBoost, the cost function \(^2\) present in the ensemble learning algorithms is mapped to an Ising-type Hamiltonian and the optimization is performed via quantum annealing. Here, we perform similar steps, but under different constraints. The basic model of ensemble learning consists in creating an ensemble of \( K \) learners \( h_k(x) \) combined in a weighted sum to perform better than each one solely. These learners are trained given a few layers and few neurons, so we call them \textit{weak-learners}. Thus, we can evaluate a real function \( f(x) \) as

\[ f(x) = \sum_{k=1}^{K} w_k h_k(x), \]

(4)

with \( x, f(x) \in \mathbb{R} \), and assuming the learners were trained beforehand, such that the parameters of their models could be omitted.

Training an ensemble model means finding the associated weights \( w_k \) for each learner. For this task, we usually minimize two terms simultaneously: a cost function \( L(w, h, x) \) and a \textit{regularization term} \( R(w) \). The cost function, chosen as a convex least-squares function,

\[ L(w, h, y) = \frac{1}{M} \sum_{m=1}^{M} \left[ \sum_{k=1}^{K} w_k h_k(x^m) - y^m \right]^2, \]

(5)

estimates the mean squared error that any regressor candidate imposes in a set of \( M \) training examples \( \{h(x^m, y^m | m = 1, \ldots, M)\} \) in relation to the true values \( y^m \) provided in this training set. The regularization, as described in \([29]\), aims to control the overfitting, and a natural choice for \( R(w) \) is an \( l_0 \)-norm penalization of \( w \), which takes the weights to zero, if possible. However, \( l_0 \)-norm regularization leads to a non-convex optimization problem, and we replaced it by \( l_2 \)-norm, so we have a convex and differentiable function.\(^3\) In this way, we must find the weights \( w \) such that,

\[ K(w, h, y) = \arg \min_w \{ J(w, h, y) \}, \]

(6)

\(^2\) The terms cost function and loss function will be used as synonyms.

\(^3\) Although the minimization of convex functions can efficiently be solved by quantum computers, here we are proposing only a proof of concept of the application of the QBoost technique to solve regression problems.
where the total cost function is defined by,

\[ J(w, h(x), y) = L(w, h(x), y) + R(\omega), \]  

(7)

\[ R(\omega) = \lambda ||w||_2^2, \]

(8)

where the terms not proportional to weights \( w_k \) were discarded, as they do not influence the minimization process. Equation (8) is already in the form of a quadratic unconstrained binary optimization (QUBO) problem [56], or a classical Ising-like Hamiltonian (see Eq. 3). We notice that there are different cost functions in the work: \( L(w, h, x) \) is the cost function (Eq. 5), \( J(w, h, y) \) is the total cost function (Eq. 8), \( H(Q, h, y) \) is the Hamiltonian (Eq. 14), which is similar to the QUBO Hamiltonian \( H_{QUBO} \) (Eq. 15), \( J(f) \) is the cost function (Eq. 19), and \( G(\theta_n, s_n) \) (Eq. 20) is the quadratic error. All these functions will be referred to as a cost function, and the appropriate equation will be indicated in each case.

Since we are interested in a regression problem, we need to write our weights as a \( R \)-bits floating-point approximation of the real value \( w_i \). Following the methodology described in [57], we apply the floating-point expansion to represent the weights \( w_k \) in Eq. (11). For any number \( \chi \in [0, 2) \), the binary representation with accuracy of \( R \) bits of resolution can be expressed by a string of bits \([Q_0Q_1Q_2\cdots Q_R]_2\), where \( Q_r \in \{0, 1\} \) is the value of the \( r \)-th bit, and the square bracket indicates the binary representation. In terms of a \( 2^{-r} \) power series, we have,

\[ \chi_k = \sum_{r=0}^{R-1} 2^{-r} Q_r^k. \]  

(9)

In order to represent the weights in a less restrictive domain, \( w_k \in [-d, 2c - d) \), we scale and shift \( \chi_k \) by,

\[ w_k = c\chi_k - d. \]  

(10)

When \( d > 0 \) and \( c > d/2 \), the \( w_k \) domain will always have a positive and negative region, and the precise value of \( c \) and \( d \) can be chosen according to the specific problem.

For the regression problem, we also added an extra constraint, \( \sum_{k=1}^{K} w_k = 1 \), to reduce one variable and force an affine mixture of the weights. Thus, replacing in Eq. (8), we have,
\[ J(w, h, y) = \sum_{k, k'=1}^{K-1} w_k w'_{k'} \left[ \frac{1}{M} \sum_{m=1}^{M} (h^m_k - h^m_{K})(h^m_{k'} - h^m_{K}) + \lambda(1 + \delta_{k,k'}) \right] \]
\[ + 2 \sum_{k=1}^{K-1} w_k \left[ \frac{1}{M} \sum_{m=1}^{M} (h^m_K - y^m_k)(h^m_{K} - h^m_k) \right] \]

(11)

Rewriting Eq. (11) using Eq. (9) and the scaling transformation of Eq. (10), the cost function becomes,

\[ J(h, y) = \sum_{k, k'=1}^{K-1} c^2 \left[ \frac{1}{M} \sum_{m=1}^{M} (h^m_k - h^m_{K})(h^m_{k'} - h^m_{K}) + \lambda(1 + \delta_{k,k'}) \right] \]
\[ \times \sum_{r=0}^{R-1} 2^{-r} Q^k_r \sum_{r'=0}^{R-1} 2^{-r'} Q^{k'}_{r'} + \sum_{k=1}^{K-1} c \left[ \frac{1}{M} \sum_{m=1}^{M} (h^m_k - h^m_K) \right] \]
\[ \times \left( (h^m_K - y^m_k) - d \sum_{k'=1}^{K-1} (h^m_{k'} - h^m_K) \right) - \lambda(1 - dK) \sum_{r=0}^{R-1} 2^{-r} Q^k_r . \]

(12)

The physical qubits in the D-Wave processor are accessed by a 1-dimensional linear index, so it is necessary to merge the \((k, r)\) indices into a single index \(l = R, R + 1, ..., KR - 1\) using,

\[ l(k, r) = kR + r , \]

(13)

with the inverse map given by \( k_l = [l/R] \) and \( r_l = l \mod R \). Now we proceed to the quantization of the variables \( Q_l \), which satisfy the eigenvalue equation \( \hat{Q}_l |Q\rangle = Q_l |Q\rangle \), where the idempotence condition is imposed \( \hat{Q}_l^2 = \hat{Q}_l \), implying that the eigenvalues \( Q_l \in \{0, 1\} \). The eigenvectors are represented by \( |Q\rangle = |Q_R\rangle \otimes |Q_{R+1}\rangle \otimes \cdots \).
... $\otimes |Q_{(KR-1)}\rangle$. Therefore, the quantization of Eq. (12) gives rise to the Hamiltonian,

$$H(Q, h, y) = \sum_{l=R, l \neq l'} c 2^{-(r_l + r_{l'})} \left[ \frac{1}{M} \sum_{m=1}^{M} (h_{k_l}^m - h_K^m)(h_{k_{l'}}^m - h_K^m) + \lambda (1 + \delta_{k_l, k_{l'}}) \right]$$

$$+ \lambda (1 + \delta_{k_l, k_{l'}}) \hat{Q}_l \hat{Q}_{l'} + \sum_{l=R}^{KR-1} c 2^{-r_l+1} \left\{ \frac{1}{M} \sum_{m=1}^{M} (h_{k_l}^m - h_K^m) \right\}$$

$$\times \left[ (h_K^m - y_m) - d \sum_{k_{l'}=1}^{K-1} (h_{k_{l'}}^m - h_K^m) + c 2^{-r_l-1} (h_{k_l}^m - h_K^m) \right]$$

$$- \lambda (1 + d K + c 2^{-r_l}) \right\} \hat{Q}_l.$$

Comparing the above equation with the QUBO Hamiltonian given by,

$$H_{\text{QUBO}} = \sum_i \alpha_i \hat{Q}_i + \sum_{i \neq j} \beta_{ij} \hat{Q}_i \hat{Q}_j,$$

we identify the coefficients of the final Hamiltonian that are the input parameters for the quantum annealing,

$$\alpha_i = c 2^{-r_l+1} \left\{ \frac{1}{M} \sum_{m=1}^{M} (h_{k_l}^m - h_K^m) \left[(h_K^m - y_m) - d \left( \sum_{k_{l'}=1}^{K-1} (h_{k_{l'}}^m - h_K^m) \right) \right] + c 2^{-r_l-1} (h_{k_l}^m - h_K^m) \right\} - \lambda (1 + d K + c 2^{-r_l}) \right\},$$

and

$$\beta_{ij} = c 2^{-(r_l+r_{l'})} \left[ \frac{1}{M} \sum_{m=1}^{M} (h_{k_l}^m - h_K^m)(h_{k_{l'}}^m - h_K^m) + \lambda (1 + \delta_{k_l, k_{l'}}) \right].$$

### 3 Hybrid algorithm

In this work, we propose a hybrid algorithm that is able to perform regression in PDEs solutions. The classical component of the algorithm consists in creating neural networks (NNs) capable of approaching the PDE solution, while the quantum component will create a boosting algorithm from the ensemble of classical NN (the weak-learners) to generate a strong NN. The quantum step of the algorithm is performed through the quantum annealing, which obtains the closest solution to the ground state of the Hamiltonian (Eq. 14), or equivalently, it finds the weights of a stronger NN.
3.1 Classical component

In order to introduce the classical component of the algorithm, let us consider a general $d$-dimensional parabolic PDE:

$$\frac{\partial}{\partial t} u(t, x) + \mathcal{L}u(t, x) = 0; \quad (t, x) \in [0, T] \times \Omega,$$

$$u(0, x) = u_0(x),$$

$$u(t, x) = g(t, x), \quad x \in \partial \Omega,$$

where $x \in \Omega \subset \mathbb{R}^d$, $t$ is time, $\mathcal{L}$ is an operator containing spatial derivatives, $u(t, x)$ is the solution of the PDE, $u_0(x)$ is the initial condition, and $g(t, x)$ is the boundary condition. This part of the algorithm performs a regression in the PDE solution through a NN, approximating the solution $u(t, x) \approx f(t, x, \theta)$, where $\theta$ is a set of real variables used to represent the internal parameters of the NN. Following [45], we use as a measure of approximation the following cost function $J(f)$,

$$J(f) = \gamma_1 \|\frac{\partial}{\partial t} f(t, x, \theta) + \mathcal{L}f(t, x, \theta)\|_{[0, T] \times \Omega, \rho_1}^2 + \gamma_2 \|f(t, x, \theta) - g(t, x)\|_{[0, T] \times \partial \Omega, \rho_2}^2 + \gamma_3 \|f(0, x, \theta) - u_0(x)\|_{\Omega, \rho_3}^2,$$

with $\|f(y)\|_{\mathcal{Y}, \rho}^2 = \int_{\mathcal{Y}} |f(y)|^2 \rho(y)dy$, $\rho(y)$ is a probability density over the domain $y \in \mathcal{Y}$ and $\gamma_i$ are weights such that $\sum_i \gamma_i = 1$. In this work, $\gamma_1 = 0.49$, $\gamma_2 = 0.01$ and $\gamma_3 = 0.5$. The three terms on the right hand side of Eq. (19) measure how well $f(t, x, \theta)$ satisfies the operators of the differential equation, boundary conditions, and initial conditions, respectively.

The NN role is to find the parameters $\theta$ which minimize the cost function $J(f)$. As $J(f) \rightarrow 0$ the NN solution approaches the PDE solution, i.e., $f(t, x, \theta) \rightarrow u(t, x)$. The advantage of this algorithm when compared to standard approaches in computational fluid dynamics is that it is not necessary to create a mesh, which may be computationally expensive. As described in [45], the algorithm consists of the following steps:

1. Generate random points $(t_n, x_n) \in [0, T] \times \Omega$, $(\tau_n, z_n) \in [0, T] \times \partial \Omega$ and $a_n \in \Omega$ according to probability densities $\rho_1$, $\rho_2$, and $\rho_3$, respectively. In this paper $\rho_1$, $\rho_2$, and $\rho_3$ are taken as the uniform probability density function.
2. Calculate the quadratic error $G(\theta_n, s_n)$ on the points drawn in 1. using the set $s_n = \{(t_n, x_n), (\tau_n, z_n), a_n\}$

$$G(\theta_n, s_n) = \gamma_1 \left[ \frac{\partial}{\partial t} f(t_n, x_n, \theta_n) + \mathcal{L}f(t_n, x_n, \theta_n) \right]^2 + \gamma_2 \left[ f(\tau_n, z_n, \theta_n) - g(\tau_n, z(n)) \right]^2 + \gamma_3 \left[ f(0, a_n, \theta_n) - u_0(a_n) \right]^2.$$

(20)
3. Update the weights $\theta_n$ using steepest gradient,

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta G(\theta_n, s_n),$$

(21)

where $\alpha_n$ is called a learning rate and decreases as $n$ increases.

4. Repeat the steps until the convergence criterion is reached $G(\theta_n, s_n) \leq \epsilon$.

The gradient $\nabla_\theta G(\theta_n, s_n)$ is an unbiased estimator of $\nabla_\theta J(f(\cdot, \theta_n))$,

$$\mathbb{E}[\nabla_\theta G(\theta_n, s_n) | \theta_n] = \nabla_\theta J(f(\cdot, \theta_n)).$$

(22)

In this way, the stochastic descent gradient algorithm will on average take steps in the downward direction of the objective function $J(f(\cdot, \theta))$, i.e., $J(f(\cdot, \theta_{n+1})) < J(f(\cdot, \theta_n))$ and therefore $\theta_{n+1}$ is a better parameter estimate than $\theta_n$.

### 3.2 Quantum component

The quantum component of the algorithm consists in performing a boosting in the ensemble learning step through the quantum annealing method. Here we propose a variant of QBoost [29–31]. Specifically, the work focuses on the study of a regression problem through an ensemble of the form $f(t, x) = \sum_{k=1}^{K} w_k h_k(t, x)$, where $(t, x) \in [0, T] \times \Omega$, $h(t, x) \in \mathbb{R}^K$ is a vector of outputs from neural networks previously trained in $(t, x)$, and $w \in \mathbb{R}^K$ is a vector of weights to be optimized such that $\sum_i w_i = 1$. We describe the algorithm as follows:

1. Validate the weak-learners produced by the classical component of the algorithm to create the training and test sets with $M$ and $N$ samples, respectively.
2. Calculate the QUBO coefficients with the training set using Eqs. (16) and (17).
3. Use the coefficients as input for D-Wave’s quantum annealers.
4. Read the binary string output from D-Wave Systems and use Eqs. (9) and (10) to reconstruct the optimal weights.
5. Construct the final output of the ensemble $f(t, x)$ with the rebuilt weights.

### 4 Results and Discussion

In order to validate our approach, we use a PDE that is well known and that has an analytical solution. The toy model chosen was that of the 1D Burgers’ equation with viscosity $v$ [58]. Consider Eq. (18) in the domain $(t, x) \in [0, T] \times [0, 2\pi]$ with the following initial and boundary conditions,

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = v \frac{\partial^2 u}{\partial x^2},$$

(23)

$$u(0, x) = -2 \frac{v}{\phi(0, x)} \frac{d\phi}{dx} + 4, \ x \in [0, 2\pi],$$

(24)

$$u(t, 0) = u(t, 2\pi), \ t \in [0, T].$$

(25)
Table 1 Weak-learners of the viscous 1D Burgers’ equation

| Weak-learner | Cost     | Time/epochs | Neurons × Layers |
|--------------|----------|-------------|------------------|
| [120,30,120,30,120] | 0.004283 | 1.619587    | 2100             |
| [10,20,30]    | 0.001574 | 0.276350    | 180              |
| [20,20,20,20,20,20] | 0.002017 | 0.284775    | 720              |
| [60,30,10,10,30,60] | 0.008058 | 1.266518    | 760              |

The weak-learner $[l_1, l_2, \ldots, l_i, \ldots, l_q]$ represents a NN with $q$ layers containing $l_i$ neurons in the $i$-th layer. The values of the cost function (20) were calculated using the test set. The time/epochs parameter represents the difficulty of training an NN.

Under these conditions, there exists an analytical solution in the form,

$$u(t, x) = -2\nu \frac{d\phi}{\phi(t, x) \, dx} + 4, \quad (t, x) \in [0, T] \times [0, 2\pi],$$

(26)

where,

$$\phi(t, x) = \exp -\frac{(x - 4t)^2}{4\nu(t + 1)} + \exp -\frac{(x + 4t - 2\pi)^2}{4\nu(t + 1)}.$$

(27)

In Table 1, we present the weak-learners corresponding to the classical component of the algorithm used to create the ensemble. More information about the generation and criterion to choose the weak-learners that form the ensemble can be found in Appendix 1. All NNs were trained 2,000 instances through random drawing of 10 points in space with $x \in [0, 2\pi]$ for each of the times $t \in \{0.0, 0.05, 0.15, 0.20, 0.30, 0.35, 0.40, 0.50\}$ for the training set and $t \in \{0.10, 0.25, 0.45\}$ for the test set, which amounts to $M = 160,000$ and $N = 60,000$ instances of training and testing, respectively. The probability densities $\rho_1$, $\rho_2$, and $\rho_3$ were obtained considering the values of the functions $u(t, x)$, $u(0, x)$, and $u(t, 0) = u(t, 2\pi)$, respectively, evaluated for each pair $(t, x)$ as defined above. We chose $\epsilon = 550$.

We ran our algorithm by solving the QUBO problem given by Eq. (15) through the D-Wave Ocean Package in three different ways: (a) using the Exact Solver function, (b) the simulated annealing, and (c) running on the DW_2000Q_6 QPU. As the Hamiltonian (Eq. 15) is diagonal in the computational basis, the Exact Solver brute force computes the energy spectrum associated with the $2^l$ possible combinations of the basis states of $l$ logical qubits. For this reason, the time to compute the ground state increases exponentially with the number of precision qubits (see Table 2). For each instance of time evolution, the simulated annealing performs 10,000 rounds with 1,000 sweeps. The $\beta$ parameter, which is proportional to the inverse of the temperature, ranges linearly from 0.1 to 4.0.\textsuperscript{4} The DW_2000Q_6 QPU operates at an average temperature of 13.5 ± 1.0 mK. As in the simulated annealing, 10,000 annealing

\textsuperscript{4} No optimization has been done in order to find better solutions via simulated annealing. Except the number of rounds, the other parameters have been kept as default.
Table 2 Values of cost function and the time needed to find the solution for different precision qubits in Eq. 14 and different methods

| R   | ES Cost | Time (s) | SA Cost | Time (s) | QA Cost |
|-----|---------|----------|---------|----------|---------|
| 3   | 0.0012255 | 0.001230 | 0.0012256 | 220.6361 | 0.0012256 |
| 4   | 0.0010381 | 0.010902 | 0.0010381 | 374.1277 | 0.0010381 |
| 5   | 0.0010023 | 0.122851 | 0.0009811 | 548.3363 | 0.0010422 |
| 6   | 0.0009964 | 1.350524 | 0.0009964 | 806.4192 | 0.0010960 |
| 7   | **0.0009944** | 14.24427 | 0.0009962 | 1069.6614 | 0.0010628 |
| 8   | – – | – | **0.0009920** | 1406.8642 | 0.0010224 |
| 9   | – – | – | – | 1686.3617 | 0.0011212 |
| 10  | – – | – | 0.0010247 | 2126.4362 | 0.0010751 |
| 11  | – – | – | 0.0010034 | 2538.7133 | **0.0010139** |
| 12  | – – | – | 0.0010067 | 3228.2340 | 0.0010382 |
| 13  | – – | – | 0.0010601 | 3440.5179 | 0.0010151 |
| 14  | – – | – | 0.0010508 | 4050.5976 | 0.0011689 |

Values in bold describe the best time for each method.

The parameter $R$ is the precision of the floating-point expansion, while the methods used are special functions of the D-Wave Ocean package, providing the exact diagonalization through the Exact Solver (ES), the classical annealing through the simulated annealing (SA), and the quantum annealing (QA) ran in the 2000Q QPU. The annealing time is the same for all levels of precision in the quantum annealing solutions processes have been undertaken for each evolution time, such that each annealing time requires 20 $\mu$s [59].

The weights of the ensemble are represented in the interval $w_k \in [-3, 3]$, which means $c = d = 3$. In Fig. 1, we show the comparison between the solution produced by our ensemble, with $\lambda = 0.0$ (the criteria for choosing $\lambda$ can be seen in Appendix 1), against the analytical solution of the 1D Burgers’ equation for the times present in the test set. It is possible to observe that the solution produced by each method is very similar to the analytical solution.

The cost function (Eq. 15) of our ensemble was evaluated in the test set, as shown in Table 2 for different number of precision qubits $R$. In all cases, the ensemble was able to perform better than all weak-learners used to compose them, see Table 1 and Fig. 2. For the simulated annealing, notice that the values of the cost function attain a minimum value of 0.0009920, which is achieved with an accuracy of eight qubits and is not reduced with additional qubits. Considering the quantum annealing we also observe a minimum in the values of the cost function of 0.0010139, which occurs for $R = 11$. For the sake of comparison with standard optimization methods, we use the stochastic gradient descent to minimize Eq. 8. After 10,000 rounds on a laptop running in double precision, we obtained the average value 0.0010200 for $L(w, h(x), y)$ with the corresponding variance $1.46 \times 10^{-5}$. We notice that this lies within one standard deviation of the value produced by the 2000Q QPU according to Table 2.

The dependence of the cost function for different number of precision qubits $R \in \{3, 4, \cdots, 14\}$ is shown in Fig. 2, obtained from QBoost method for the test set on the
Fig. 1 Analytical solution $U_a(x)$ (lines) and the QBoost solution $U_{qb}(x)$ (markers) with $R = 6$ of the test set for the 1D viscous Burgers’ equation. From left to right, we have curves for $t = 0.1$, $t = 0.25$ and $t = 0.45$, respectively. The solutions were obtained using three different functionalities of the D-Wave Ocean package: a) exact diagonalization of Hamiltonian (Eq. 14) through the Exact Solver, b) classical solution through the simulated annealing, and c) quantum solution through the 2000Q system.

first 11 energy levels of the final Hamiltonian (Eq. 14). For sake of comparison, the cost function calculated from the best weak-learner is also shown (cf. the horizontal line). As a result, the QBoost with $R \geq 3$ performs better than the best weak-learner for all energy levels displayed. We also observe that as we increase the number of precision qubits, the values of the cost function become closer to each other. Indeed, this demonstrates that even solutions that did not reach the ground state of the Hamiltonian can be considered valid, as the errors associated with these states are very close, within a predefined tolerance, $0.0005$.

The simulation using the Exact Solver has a limitation in relation to the number of precision qubits owing to the amount of memory required for its execution, which increases exponentially. We were able to achieve only $R = 7$ on our personal computer of 32 GB RAM. Note that as we increase $R$, the computational time increased approximately by one order of magnitude. The time of the simulated annealing grows smoothly. However, it starts from a value five orders of magnitude higher than the time spent by the Exact Solver.
Although we have obtained good results even for a small number of precision qubits, we list the following caveats:

- We are minimizing the cost function (Eq. 8), which consists of finding the values of the weights \( w_k \) over the real domain for a quadratic function. This is a least-squares problem, which can be solved efficiently by a classical computer [60]. However, even in this case it is fair to search for polynomial speedups through quantum annealers.

- The continuous variables were expressed using float-pointing expansion [57]. This methodology results in an exponential narrowing of the energy gap of the final Hamiltonian as the number of precision qubits increases. We see this behavior from Eq. (12) in the following manner. As the variables \( Q_k^r \in \{0, 1\} \) and the small contributions to the equation come from the highest powers \( r \) of \( 2^{-r} \), we observe that the energy difference between adjacent levels will always be multiplied by factors that decrease exponentially with the precision of the continuous variables, see Fig. 2. In spite of this undesired property for quantum computing based on adiabatic evolution, in our example this seems not to interfere significantly in the problem solution, as can be seen in Table 2. In addition, as we are looking for weights \( w_k \) to compose the ensemble of weak-learners, the precision of \( w_k \) could otherwise be compensated by the addition of more weak-learners.

It is important to mention that a proper comparison between the QBoost for regression problems and standard classical formulations applied to solve the 1D Burgers’ equation, given the current stage of development of quantum annealers, is not quite fair. For instance, with a finite difference algorithm applied to the 1D Burgers’ equation we obtained a mean square error of \( 1.51 \times 10^{-8} \) for the same test set, \( t \in \{0.10, 0.25, 0.45\} \). In spite of obtaining a result that is five orders of magnitude better, the main shortcomings of quantum annealers are the high level of noise [61] and a limited number of qubits [62], which impacts the precision of the quantum solution. While a laptop
running with double precision uses 64 bits, the maximum number of precision qubits in our example is 14. For this reason, we are restricted to a proof of concept of our method.

5 Conclusion

An adaptation of the QBoost algorithm to solve the regression problem using floating-point approximations to represent real variables has been successfully applied to solve the 1D Burgers’ equation with viscosity. Our framework did not result in overfitting or underfitting and is in very good agreement with the analytical solution (see Fig. 1).

The floating-point approximation should be used carefully, since the energy gap decreases rapidly with increasing precision, although the accepted solutions are very close to the ground state, and therefore, are good approximations to the optimal solution. Another potential issue pertains to the connectivity between qubits, which may limit the number of weak-learners used in the ensemble.

Supplementary information

The data and codes used to produce the results in this paper are publicly available at https://bitbucket.org/quantumcomp/qboost.git

Acknowledgements We acknowledge the financial support by Brazilian agencies CAPES, CNPq, and INCT-IQ (National Institute of Science and Technology for Quantum Information).

Appendix A: Weak-learners

In Table 3, we list all weak learners produced for this work. The lines in bold represent those chosen to compose the ensemble. We can see that there is no predetermined pattern relating architecture, number of neurons and hidden layers with the cost function. The choice of which weak-learner to compose the ensemble followed the criterion: (i) the ensemble should include the weak-learner with the overall best performance, such that our framework can be validated in relation to the performance of the ensemble; and (ii) other weak-learners should exhibit medium to poor performance, in order to demonstrate that the use of weak-learners with poor performance can also generate good ensembles.

Appendix B: Regularization term

The dependence of the cost function (Eq. 5) on the regularization parameter $\lambda$ is shown in Fig. 3. We see that any increase in the regularization parameter within the range shown results in an increase of the cost function. Therefore, we conclude that the best result occurs when the regularization term is null. Our conclusion is supported by the fact that we did not observe the presence of overfitting or underfitting in Fig. 1.
Table 3  Weak-learners of the viscous 1D Burgers’ equation

| Weak-learner | Cost      | Time/epochs | Neurons x Layers |
|--------------|-----------|-------------|------------------|
| [10,30,60,90,110,150] | 0.015924  | 0.973962    | 2700             |
| [180,90,30,30,90,180] | 0.011367  | 1.027333    | 3600             |
| [30,90,180,180,90,30] | 0.010340  | 1.380745    | 3600             |
| [60,180,60,180,60,180] | 0.010368  | 1.274568    | 4320             |
| [20,20,20,20,20,20]  | **0.002017** | **0.284775** | **720**         |
| [10,20,30,40,50,60]  | 0.004267  | 0.473657    | 1260             |
| [60,50,40,30,20,10]  | 0.004805  | 0.526772    | 1260             |
| [60,30,10,10,30,60]  | **0.008058** | **1.266518** | **760**         |
| [10, 30, 60, 60, 30, 10] | 0.007132  | 1.788494    | 760              |
| [20, 20, 60, 60, 20, 60] | 0.002611  | 1.511132    | 1440             |
| [150,90,30,90,150]  | 0.011307  | 1.568148    | 2550             |
| [30,90,150,90,30]   | 0.008128  | 1.767931    | 1950             |
| [30, 120, 30, 120, 30] | 0.007504  | 1.046415    | 1650             |
| [120, 30, 120, 30, 120] | **0.004283** | **1.619587** | **2100**        |
| [10, 30, 50, 70, 90] | 0.006257  | 1.249093    | 1250             |
| [90, 70, 50, 30, 10] | 0.009561  | 1.480899    | 1250             |
| [25, 25, 25, 25, 25] | 0.003077  | 0.523823    | 625              |
| [15, 15, 15, 15, 15] | 0.002436  | 0.304417    | 375              |
| [150, 60, 60, 150]  | 0.012856  | 0.657789    | 1680             |
| [60, 150, 150, 60]  | 0.013863  | 0.809011    | 1680             |
| [90, 30, 90, 30]   | 0.001743  | 0.456623    | 960              |
| [30, 90, 90, 30]   | 0.006373  | 0.472136    | 960              |
| [90, 60, 30, 10]   | 0.005259  | 0.418490    | 760              |
| [10, 30, 60, 90]  | 0.013063  | 0.399743    | 760              |
| [30, 30, 30, 30]   | 0.003695  | 0.312810    | 480              |
| [60, 10, 10, 60]  | 0.002284  | 0.308086    | 560              |
| [10, 60, 60, 10]  | 0.003263  | 0.337634    | 560              |
| [45, 10, 45, 10]  | 0.005858  | 0.245707    | 440              |
| [10, 45, 10, 45]  | 0.005785  | 0.256790    | 440              |
| [40, 30, 20, 10]  | 0.002407  | 0.279718    | 400              |
| [10, 20, 30, 40]  | 0.002154  | 0.274698    | 400              |
| [10, 10, 10, 10]  | 0.002184  | 0.193450    | 160              |
| [150, 60, 150]  | 0.002990  | 0.805101    | 1080             |
| [60, 150, 60]   | 0.013825  | 1.251153    | 810              |
| [10,45,90]       | 0.004367  | 0.911909    | 435              |
Table 3 continued

| Weak-learner | Cost   | Time/epochs | Neurons x Layers |
|--------------|--------|-------------|------------------|
| [90, 45, 10] | 0.005408 | 0.603616    | 435              |
| [40, 40, 40] | 0.002905 | 0.469319    | 360              |
| [90, 30, 90] | 0.005547 | 1.179521    | 630              |
| [30, 90, 30] | 0.003455 | 1.066604    | 450              |
| [10, 20, 30] | **0.001574** | **0.276350** | 180              |
| [30, 20, 10] | 0.011371 | 0.204170    | 180              |
| [10, 10, 10] | 0.001884 | 0.158107    | 90               |

The weak-learner \([l_1, l_2, \cdots, l_i, \cdots, l_q]\) represents a NN with \(q\) layers containing \(l_i\) neurons in the \(i\)-th layer. The values of the cost function (20) were calculated using the test set. The time/epochs parameter represents the difficulty of training an NN. The lines in bold represent the weak-learners selected to compose the ensemble.

Fig. 3 Cost function (Eq. 5) as a function of the regularization parameter \(\lambda\)

References

1. Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., Zdeborová, L.: Machine learning and the physical sciences. Rev. Mod. Phys. 91(4), 1 (2019). https://doi.org/10.1103/revmodphys.91.045002
2. Hermann, J., Schätzle, Z., Noé, F.: Deep-neural-network solution of the electronic schrödinger equation. Nat. Chem. 12(10), 891–897 (2020). https://doi.org/10.1038/s41557-020-0544-y
3. Kieferová, M., Wiebe, N.: Tomography and generative training with quantum Boltzmann machines. Phys. Rev. A 96(6), 1 (2017). https://doi.org/10.1103/physreva.96.062327
4. Torlai, G., Mazzola, G., Carrasquilla, J., Troyer, M., Melko, R., Carleo, G.: Neural-network quantum state tomography. Nat. Phys. 14(5), 447–450 (2018). https://doi.org/10.1038/s41567-018-0048-5
5. August, M., Ni, X.: Using recurrent neural networks to optimize dynamical decoupling for quantum memory. Phys. Rev. A 95(1), 1 (2017). https://doi.org/10.1103/physreva.95.012335
6. Canabarro, A., Fanchini, F.F., Malvezzi, A.L., Pereira, R., Chaves, R.: Unveiling phase transitions with machine learning. Phys. Rev. B 100, 045129 (2019). https://doi.org/10.1103/PhysRevB.100.045129
7. Carrasquilla, J., Melko, R.G.: Machine learning phases of matter. Nat. Phys. 13(5), 431–434 (2017). https://doi.org/10.1038/nphys4035
8. Dral, P.O.: Quantum chemistry in the age of machine learning. J. Phys. Chem. Lett. 11(6), 2336–2347 (2020). https://doi.org/10.1021/acs.jpclett.9b03664
9. Hezaveh, Y.D., Levasseur, L.P., Marshall, P.J.: Fast automated analysis of strong gravitational lenses with convolutional neural networks. Nature 548(7669), 555–557 (2017). https://doi.org/10.1038/nature23463

10. Agresti, I., Viggianiello, N., Flamini, F., Spagnolo, N., Crespi, A., Osellame, R., Wiebe, N., Sciarrino, F.: Pattern recognition techniques for boson sampling validation. Phys. Rev. X 9(1), 1 (2019). https://doi.org/10.1103/physrevx.9.011013

11. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org

12. Yoo, S., Bang, J., Lee, C., Lee, J.: A quantum speedup in machine learning: finding anN-bit Boolean function for a classification. New J. Phys. 16(10), 103014 (2014). https://doi.org/10.1088/1367-2630/16/10/103014

13. Cai, X.-D., Wu, D., Su, Z.-E., Chen, M.-C., Wang, X.-L., Li, L., Liu, N.-L., Lu, C.-Y., Pan, J.-W.: Entanglement-based machine learning on a quantum computer. Phys. Rev. Lett. 114(11), 1 (2015). https://doi.org/10.1103/physrevlett.114.110504

14. Yu, C.-H., Gao, F., Lin, S., Wang, J.: Quantum data compression by principal component analysis. Quant. Inf. Process. 18(8), 1 (2019). https://doi.org/10.1007/s11128-019-2364-9

15. Pepper, A., Tischler, N., Pryde, G.J.: Experimental realization of a quantum autoencoder: the compression of qutrits via machine learning. Phys. Rev. Lett. 122(6), 1 (2019). https://doi.org/10.1103/physrevlett.122.060501

16. Huang, C.-J., Ma, H., Yin, Q., Tang, J.-F., Dong, D., Chen, C., Xiang, G.-Y., Li, C.-F., Guo, G.-C.: Realization of a quantum autoencoder for lossless compression of quantum data. Phys. Rev. A 102(3), 1 (2020). https://doi.org/10.1103/physreva.102.032412

17. Dang, Y., Jiang, N., Hu, H., Ji, Z., Zhang, W.: Image classification based on quantum k-nearest-neighbor algorithm. Quant. Inf. Process. 17(9), 1 (2018). https://doi.org/10.1007/s11128-018-2004-9

18. Wang, Y., Wang, R., Li, D., Ady-Gyamfi, D., Tian, K., Zhu, Y.: Improved handwritten digit recognition using quantum k-nearest neighbor algorithm. Int. J. Theor. Phys. 58(7), 2331–2340 (2019). https://doi.org/10.1007/s10773-019-04124-5

19. Farhi, E., Gutmann, S.: Quantum computation and decision trees. Phys. Rev. A 58(2), 915–928 (1998). https://doi.org/10.1103/physreva.58.915

20. Lu, S., Braunstein, S.L.: Quantum decision tree classifier. Quantum Inf. Process. 13(3), 757–770 (2013). https://doi.org/10.1007/s11128-013-0687-5

21. Lloyd, S., Weedbrook, C.: Quantum generative adversarial learning. Phys. Rev. Lett. 121(4), 1 (2018). https://doi.org/10.1103/physrevlett.121.040502

22. Schuld, M., Petruccione, F.: Quantum ensembles of quantum classifiers. Sci. Rep. 8, 1 (2018). https://doi.org/10.1038/s41598-018-20403-3
33. Abbas, A., Schuld, M., Petruccione, F.: On quantum ensembles of quantum classifiers. Quant. Mach. Intell. 2(1), 1 (2020). https://doi.org/10.1007/s42484-020-00018-6
34. Salcedo-Sanz, S., Ortiz-García, E.G., Pérez-Bellido, Á.M., Portilla-Figuera, A., Prieto, L.: Short term wind speed prediction based on evolutionary support vector regression algorithms. Expert Syst. Appl. 38(4), 4052–4057 (2011). https://doi.org/10.1016/j.eswa.2010.09.067
35. Salcedo-Sanz, S., Ghamisi, P., Piles, M., Werner, M., Cuadra, L., Moreno-Martínez, A., Izquierdo-Verdiguier, E., Muñoz-Marí, J., Mosavi, A., Camps-Valls, G.: Machine learning information fusion in earth observation: A comprehensive review of methods, applications and data sources. Inf. Fus. 63, 256–272 (2020). https://doi.org/10.1016/j.inffus.2020.07.004
36. Wu, Y., Tan, H., Peter, J., Shen, B., Ran, B.: Short-term traffic flow prediction based on multilinear analysis and k-nearest neighbor regression. In: CICTP 2015. American Society of Civil Engineers, Beijing (2015).
37. Zhang, L., Alharbe, N.R., Luo, G., Yao, Z., Li, Y.: A hybrid forecasting framework based on support vector regression with a modified genetic algorithm and a random forest for traffic flow prediction. Tsinghua Sci. Technol. 23(4), 479–492 (2018). https://doi.org/10.26599/tst.2018.9010045
38. Lou, S., Li, D.H.W., Lam, J.C., Chan, W.W.H.: Prediction of diffuse solar irradiance using machine learning and multivariable regression. Appl. Energy 181, 367–374 (2016). https://doi.org/10.1016/j.apenergy.2016.08.093
39. Alizamir, M., Kim, S., Kisi, O., Zounemat-Kermani, M.: A comparative study of several machine learning based non-linear regression methods in estimating solar radiation: Case studies of the USA and turkey regions. Energy 197, 117239 (2020). https://doi.org/10.1016/j.energy.2020.117239
40. Zhang, J., Zhao, X.: Machine-learning-based surrogate modeling of aerodynamic flow around distributed structures. AIAA J. 1, 1–12 (2020). https://doi.org/10.2514/1.j059877
41. Dupuis, R., Jouhaud, J.-C., Sagaout, P.: Surrogate modeling of aerodynamic simulations for multiple operating conditions using machine learning. AIAA J. 56(9), 3622–3635 (2018). https://doi.org/10.2514/1.j056405
42. Andrés, E., Salcedo-Sanz, S., Monge, F., Pérez-Bellido, A.M.: Efficient aerodynamic design through evolutionary programming and support vector regression algorithms. Expert Syst. Appl. 39(12), 10700–10708 (2012). https://doi.org/10.1016/j.eswa.2012.02.197
43. Richmond, M., Sobey, A., Pandit, R., Kolios, A.: Stochastic assessment of aerodynamics within offshore wind farms based on machine-learning. Renewable Energy 161, 650–661 (2020). https://doi.org/10.1016/j.renene.2020.07.083
44. Umethani, N., Bickel, B.: Learning three-dimensional flow for interactive aerodynamic design. ACM Trans. Gr. 37(4), 1–10 (2018). https://doi.org/10.1145/3197517.3201325
45. Sirignano, J., Spiliopoulos, K.: DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. 375, 1339–1364 (2018). https://doi.org/10.1016/j.jcp.2018.08.029
46. Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V.M., Guo, H., Handia, K., Zhuang, X., Rabczuk, T.: An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Comput. Methods Appl. Mech. Eng. 362, 112790 (2020). https://doi.org/10.1016/j.cma.2019.112790
47. Ranade, R., Hill, C., Pathak, J.: DiscretizationNet: a machine-learning based solver for Navier–Stokes equations using finite volume discretization. Comput. Methods Appl. Mech. Eng. 378, 113722 (2021). https://doi.org/10.1016/j.cma.2021.113722
48. Raissi, M., Karniadakis, G.E.: Hidden physics models: machine learning of nonlinear partial differential equations. J. Comput. Phys. 357, 125–141 (2018). https://doi.org/10.1016/j.jcp.2017.11.039
49. Regazzoni, F., Dedè, L., Quarteroni, A.: Machine learning for fast and reliable solution of time-dependent differential equations. J. Comput. Phys. 397, 108852 (2019). https://doi.org/10.1016/j.jcp.2019.07.050
50. Albash, T., Lidar, D.A.: Adiabatic quantum computation. Rev. Mod. Phys. 90(1), 1 (2018). https://doi.org/10.1103/revmodphys.90.015002
51. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse ising model. Phys. Rev. E 58(5), 5355–5363 (1998). https://doi.org/10.1103/physreve.58.5355
52. Kato, T.: On the adiabatic theorem of quantum mechanics. J. Phys. Soc. Jpn. 5(6), 435–439 (1950)
53. Messiah, A.: Quantum Mechanics, vol. II. North-Holland, Amsterdam (1962)
54. Sarandy, M.S., Wu, L.-A., Lidar, D.A.: Consistency of the adiabatic theorem. Quant. Inf. Process. 3(6), 331–349 (2004)
55. D-Wave: Technical Description of the D-Wave Quantum Processing Unit (2021)
56. Glover, F., Kochenberger, G.: A tutorial on formulating qubo models. arXiv preprint arXiv:1811.11538 (2018)
57. Rogers, M.L., Singleton, R.L.: Floating-point calculations on a quantum annealer: division and matrix inversion. Front. Phys. 8, 1 (2020). https://doi.org/10.3389/fphy.2020.00265
58. Burgers, J.M.: A Mathematical Model Illustrating the Theory of Turbulence. Advances in Applied Mechanics, vol. 1, pp. 171–199. Elsevier (1948). https://doi.org/10.1016/S0065-2156(08)70100-5
59. Inc, D.-W.S.: QPU-Specific Physical Properties: DW_2000Q_6 (2019)
60. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
61. Zaborniak, T., de Sousa, R.: Benchmarking Hamiltonian noise in the d-wave quantum annealer. IEEE Trans. Quant. Eng. 2, 1–6 (2021)
62. Pollachini, G.G., Salazar, J.P.L.C., Góes, C.B.D., Maciel, T.O., Duzzioni, E.I.: Hybrid classical-quantum approach to solve the heat equation using quantum annealers. Phys. Rev. A 104(3), 032426 (2021). https://doi.org/10.1007/s11128-013-0687-5

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

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.