ADVANCED QUANTUM TECHNOLOGIES
www.advquantumtech.com

Evolutionary Computation for Adaptive Quantum Device Design

*Luke Mortimer, Marta P. Estarellas, Timothy P. Spiller, and Irene D'Amico**

As noisy intermediate-scale quantum (NISQ) devices grow in number of qubits, determining good or even adequate parameter configurations for a given application, or for device calibration, becomes a cumbersome task. An evolutionary algorithm is presented here which allows for the automatic tuning of the parameters of any arrangement of coupled qubits, to perform a given task with high fidelity. The algorithm's use is exemplified with the generation of schemes for the distribution of quantum states and the design of multi-qubit gates. The algorithm is demonstrated to converge very rapidly, yielding unforeseeable designs of quantum devices that perform their required tasks with excellent fidelities. Given these promising results, practical scalability, and application versatility, the approach has the potential to become a powerful technique to aid the design and calibration of NISQ devices.

1. Introduction

Quantum technologies have already extensively manifested their potential to impact a wide spectrum of fields. The first proof-of-principle demonstration of quantum advantage in terms of computational power, also known as quantum supremacy,[1] has already been achieved,[2] satellite-based quantum-protected key sharing (quantum key distribution, QKD) has been realized,[3] and early prototypes of a first quantum internet are starting to emerge.[4] As the qubit-number and complexity of quantum technology devices increases, so does the number of their relevant parameters and, correspondingly, the size of the parameter space to investigate. Consequently, calibrating the devices and/or determining parameter values suitable to perform a desired task is becoming a very complex challenge.

L. Mortimer, Prof. T. P. Spiller, Prof. I. D'Amico
Department of Physics
University of York
York, YO10 5DD, UK
E-mail: irene.damico@york.ac.uk
Dr. M. P. Estarellas
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/qute.202100013

DOI: 10.1002/qute.202100013

At the same time, the application range is so varied that there is no established preferred physical hardware for such early devices and hybridized approaches toward technologies are currently being undertaken.[5] With quantum devices being so diverse and heterogeneous, spin networks form a very convenient mathematical model, able to capture the quantum dynamics of any arrangement of two-level quantum systems coupled to each other, independent of the actual physical implementation.[6] The specificity of each physical system is instead captured by the network topology, couplings and the energy scale of the parameters. Because of this, the simulation of quantum chips under the spin network formalism has proven to be a useful test-bed for the study and design of new quantum hardware and its applications.

Spin networks have been engineered to allow for quantum state transfer,[7–11] to present topologically protected states,[12–14] or to act as quantum gates,[15–21] among a spectrum of applications. However, to achieve a desired dynamical behavior, the ability to choose a suitable network topology is required, as well as calibration of the system parameters, such as the interaction energies, or couplings, between the different nodes (qubits) of the network. While available technology already allows for excellent control of such parameters in the laboratory,[2] determining suitable tuning, as well as an appropriate network topology, in the design of quantum devices is a highly non-trivial task for complex systems beyond toy-models. The number of possibilities and combinations is so vast that determining good or even adequate solutions for a given task can be not only cumbersome, but also counter-intuitive.

To circumvent this, for the design of quantum devices via the engineering of spin networks, we here propose the use of tools common in evolutionary computation.[22] This computational paradigm was originally proposed in the 1970s with the idea of using concepts of natural evolution to solve hard computational tasks in optimization, design, and modeling.[23] Since then, it has proven to be a powerful tool for problems that do not necessarily require optimal solutions, but instead can utilize appropriate approximations to these. To this purpose, genetic algorithms are one of the most popular techniques and have widely been applied to solve large engineering problems, ranging from antenna design,[24] and complex aerodynamic modeling[25] to the improvement of artificial neural networks[26] and the automatic identification of analytical equations underlying physics phenomena.[27] The design of such systems is

characterized by the number and complexity of their degrees of freedom, something that makes the number of possible configurations exponentially large. Genetic algorithms are highly parallelizable meta-heuristic optimization techniques able to efficiently cover such large search spaces and thus find approximate solutions.[28] Quantum systems presents similarities to these problems in terms of complexity, making genetic methods promising candidates to automate the search for appropriate design solutions. To date, such methods have been scarcely used in this field, with only one example to our knowledge.[29]

Here, we design a genetic algorithm capable of identifying optimal tuning parameters of a spin network to achieve any given quantum information task. Our algorithm is general and could be applied to any given problem of this sort. To exemplify its use, here we focus on two tasks: the engineering of quantum devices for quantum state distribution and the design of multi-qubit gates.

We not only demonstrate that the proposed automated technique may find new system configurations that were previously unknown, but we also show that it can do this very rapidly. Machine learning algorithms have recently proven useful for the tuning of semiconductor quantum devices[30,31] with a runtime of approximately 70 min for the tuning of eight experimental parameters (gate voltages of a double quantum dot). Our investigations show our approach based on genetic algorithms to be a promising alternative for the tuning and identification of unforeseeable designs of quantum devices, with examples of runtimes as short as 5 s to optimize 10 model parameters up to a fidelity of 99.7%.

2. Spin Network Model

We consider a general spin network of N sites (also referred to as spins, nodes or qubits) that can be described by the following time-independent XXZ-Heisenberg Hamiltonian:

$$\hat{H} = \sum_{i<j} J_{ij}\left(|1\rangle\langle 0|_i \otimes |0\rangle\langle 1|_j + h.c.\right) + \sum_{i=1}^{N} \epsilon_i |1\rangle\langle 1|_i + \sum_{i<j} \alpha J_{ij}\left(|1\rangle\langle 1|_i \otimes |1\rangle\langle 1|_j\right) \quad (1)$$

with J_{ij} being the real-valued coupling between sites i and j, and ϵ_i the on-site energies. In our encoding, we consider the injection of an excitation to be the creation of a spin "up", $|1\rangle$, in a system that has initially been prepared to have all the spins "down", $|0\rangle$. The interaction term is proportional to the constant dimensionless scaling factor α. For $\alpha \neq 0$, this term represents the interaction energy between two excitations[32] and thus affects the dynamics of subspaces containing at least two excitations. The topology of the network is defined by the non-zero elements of J_{ij}. The values of J_{ij} and of ϵ_i are the targets of the optimization. Throughout this paper, $J_{\max}$ indicates the maximum value of the couplings for a given system.

Once the parameters are set, we obtain the eigenvectors and eigenvalues of the Hamiltonian matrix by direct diagonalization. Then any chosen initial state is decomposed into the eigenvectors, which are each evolved via the unitary operator $U = e^{-\frac{i}{\hbar}E_i t}$, with E_i being the corresponding eigenvalue. This allows us to obtain the overall state of the system at any time without loss of accuracy.

We use the fidelity as a measure to assess how close the state of the system at a given time $|\Psi(t)\rangle$ is to the specific target state $|\Psi_{\text{target}}\rangle$ required for our task. The fidelity $F(t)$ ranges between zero and unity, with maximum fidelity representing a perfect overlap between target and actual state:

$$F(t) = |\langle \Psi_{\text{target}}|\Psi(t)\rangle|^2 \quad (2)$$

3. Genetic Algorithm

General genetic algorithms rely on the evaluation of different parametrizations of a system's degrees of freedom to perform a given task. This evaluation is done through the calculation of a "fitness" score, which indicates how close a given parametrization is to optimality. In our implementation, the parameters to be tuned are the set of non-zero coupling energies of Equation (1).

The flowchart in **Figure 1** shows a schematic of the route followed by our genetic algorithm. The first and most critical step to design a genetic algorithm is to define a proper structure of what is called the "genome." As in DNA, the genome is represented as a string containing the mutable information of a system; this is, its degrees of freedom. The algorithm starts with a set of different genomes being evaluated according to a fitness function. After this evaluation, the better genomes are favored to combine ("crossover") with other successful genomes, based on their fitness scores, to form the next generation. After crossing-over two genomes, a random modification is then made to the resulting genome to allow for new and unique solutions to be found.[33]

3.1. A Genome For a Spin Network

In order to properly utilize the features of a genetic algorithm, it is important to find an effective representation of the spin network in a standardized notation, so that it can be easily passed as an argument to the various functions of the program. As such, it should contain all of the system's relevant information, while also being easy to store, modify, and transfer. The genome we chose to use here is represented as a fixed-size linear string of standard ASCII characters, split into various sections representing information useful for the different functions. There also exist some optional features that can be added into the string to specify more unique requirements.

In our genome, letters are used to represent the sites and their corresponding single-excitation basis vectors, while couplings energies are represented as integers. For example, *AB*500 would represent a coupling strength of 500 between sites A and B of a spin network, relative to any other specified couplings. A bra-ket $\langle \ldots | \ldots \rangle$ at the start specifies the initial and target states of the system's protocol, which is the information necessary to evaluate the performance of such a genome through its fidelity (Equation (2)). In this bra-ket, two letters placed adjacent are treated as the tensor product of the two single excitation basis vectors, for instance $\langle AB| = \langle 11|_{AB}$, thus allowing the use of multiple excitation subspaces. Superpositions can be described through

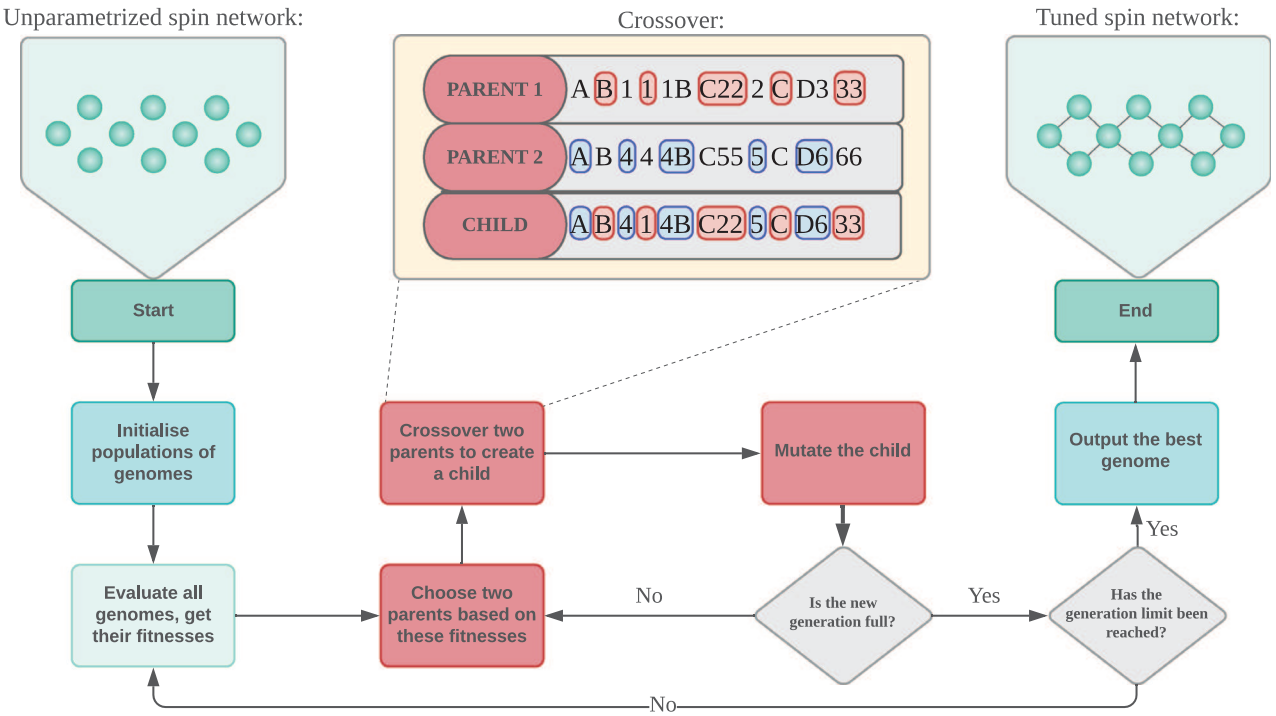

Figure 1. Schematic representation of our genetic algorithm. In the central panel, we present a diagram displaying the method used to crossover two genomes. For each character in the child genome, the corresponding character from one of the parents is used, with each having equal probability.

addition, with an optional phase (-1, i, $(1+2i)$, etc.), such that $|A+iB\rangle = \frac{1}{\sqrt{2}}(|10\rangle_{AB} + i\,|01\rangle_{AB})$. Note that in the algorithm, all state vectors are automatically normalized, so normalization factors are left out from the genome.

Two simple genomes representing a 3- and a 4-spin network are given in **Figure 2a** and 2b, respectively. In Figure 2a, an initial excitation is injected at site A, $|\Psi(0)\rangle = |100\rangle_{ABC}$, and then, at each time within a chosen set, the fidelity is evaluated against the target state $|\Psi_{\text{target}}\rangle = |001\rangle_{ABC}$. This genome specifies two coupling energies with relative values of 500 (between A and B) and 500 (between B and C), thus representing a uniform chain.

In Figure 2b, a more complicated example is presented. This genome represents a small spin network aiming to generate an entangled state between sites C and D, $|\Psi_{\text{target}}\rangle = |00\rangle_{AB} \otimes \frac{1}{\sqrt{2}}(|10\rangle_{CD} + |01\rangle_{CD})$, when a single excitation is injected at site A, $|\Psi(0)\rangle = |1000\rangle_{ABCD}$.

There are also some more niche genome features. A target time can be specified anywhere in the genome with syntax "@12.40" to force the algorithm to optimize the dynamics of the system to reach the target state at, for instance, $t_f = 12.40/J_{\max}$. This substantially increases the speed of the algorithm since only a single-point calculation corresponding to that specific time point is required, rather than evaluating the full dynamics (and searching) over a time window. Although this allows for faster optimization, it removes any flexibility in the time and thus should only be used if the transfer time is known, or is to be specified rather than allowed to vary. As such, it is often useful to perform a short optimization without this feature first, to see the time that the system naturally evolves toward, and then perform a subsequent optimization specific to that time.

Nonuniform on-site energies can also be specified, achieved by including a repeated-letter coupling (such as *AA*650). Negative couplings are also allowed, which could be of interest when evaluating systems of different magnetic order, such as antiferromagnetic lattices.[34] These are achieved through specifying a coupling with the letters in reverse-alphabetical order, such that *BA*500 represents −500. Such notation is used to keep genome length constant and concise. There also exists optional notation related to the visualization of the genome.[35]

3.2. Fitness

For the evaluation of the genomes, one needs to define first a fitness function which takes a genome string as an input and returns its fitness (normalized to be a number between 0 and 100), indicating how well such a genome satisfies the protocol or device requirements. This function combines various factors such as the maximum fidelity between the evolving state and the target state, along with the time at which that state is reached, all scaled by customizable parameters.

We chose the function to be exponential in order to give any genome which is mutated positively a more significant boost in fitness, such that a small increase in maximum fidelity results in a much greater fitness score, and long times taken to reach the maximum fidelity are penalized. The overall equation for this fitness function is given as

$$f(F_{\max}, t_f) = 100\,\exp\left(a(F_{\max}-1)\right)\,\exp\left(bt_f J_{\max}\right) \tag{3}$$

where $F_{\max}$ is the maximum fidelity, t_f is the time in units of $1/J_{\max}$ to reach $F_{\max}$, and a and b are suitably chosen scaling

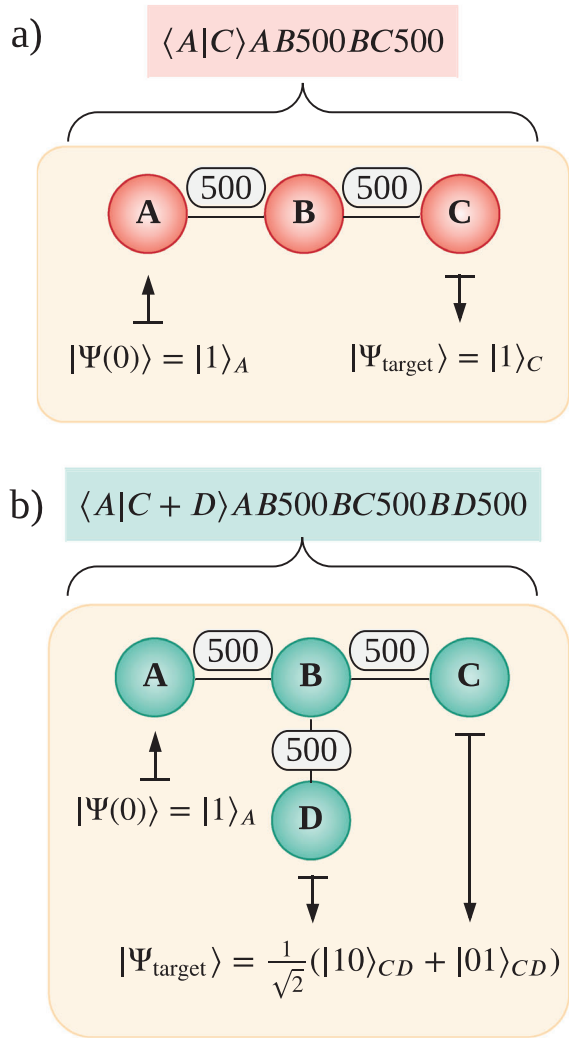

Figure 2. Examples of the construction of a genome in our algorithm. a) Uniformly coupled ($J = 500$) N=3 spin chain with initial state $|\Psi(0)\rangle = |100\rangle_{ABC}$ and target state $|\Psi_{\text{target}}\rangle = |001\rangle_{ABC}$. b) Uniformly coupled ($J = 500$) spin network with initial state $|\Psi(0)\rangle = |1000\rangle_{ABCD}$ and target state $|\Psi_{\text{target}}\rangle = |00\rangle_{AB} \otimes \frac{1}{\sqrt{2}}(|10\rangle_{CD} + |01\rangle_{CD})$. Note that in the diagram, factorizable $|0\rangle$ states are left out for convenience.

factors. For the fidelity, the fitness function clearly grows exponentially with $F_{\max}$, but once $1 - F_{\max}$ decreases below $1/a$, this strong $F_{\max}$-dependence flattens off. We have therefore found it effective to use $a = 10$, since for many systems a fidelity greater than 90% is then reached quickly, with the remainder of the optimisation spent on fine-tuning the system $F_{\max}$ toward unity, or 100%. For the time t_f, with $b < 0$, the second exponential term in the fitness function clearly encourages vanishingly small values of t_f, but does not significantly penalize the time until $t_f J_{\max} \approx 1/|b|$. We have therefore found it effective to use $b = -0.001$ for the calculations presented here. Increasing $|b|$, for example to $b = -0.01$, would place more emphasis on short t_f in the fitness function, compared to the fidelity behavior.

In any given run, the maximum fidelity is found by searching the dynamics of the system over a search window, by default between 0 and $20/J_{\max}$ divided into 100 increments. The time-window width, $20/J_{\max}$, was chosen to allow the dynamics to reach peak fidelity in spin networks of the size here considered. This time-window is in fact about one order of magnitude larger than $1/J_{\max}$, the typical time an excitation would take to tunnel through a link with maximum coupling.

Time window and number of increments were chosen to provide a balance between search resolution and performance, and can be easily changed if larger or more precise search regions are needed, perhaps for transfer over very long chains or for rapidly fluctuating fidelities, respectively. For all of the examples in this paper, however, this range is able to capture the maximum fidelity without requiring too many increments.

A fitness score of 100 for a spin-chain designed as a state-transfer device would thus mean that the transfer has been unrealistically achieved with zero waiting time ($t_f = 0$) and perfect fidelity ($F(t_f) = 1$), while a value approaching 0 would suggest either no information transfer at all or that it takes so long in time that it is not an effective solution. This fitness is then used to determine the likelihood that the features contained within a certain genome will continue through the generations.

3.3. Crossover and Mutation

In order to create the next generation, genomes from the previous generation are selected with a probability proportional to their fitness score. When two genomes are selected, they are combined in a process known as crossover. In this particular implementation, crossover involves iterating over the number of characters in the parent genomes and for each genome position randomly choosing (with equal probability) one of the parents from which to take the character, as shown visually in the crossover panel of Figure 1. Note that since both genomes share the same letter order, it is only the coupling values which change.

This new genome, also referred to as the child, is then "mutated" by increasing or decreasing one of its couplings by a random integer less than or equal to the maximum mutation size, μ, generated uniformly. The new coupling is capped between 0 and the highest possible coupling for that genome, unless negative couplings are explicitly allowed. This μ begins at some initial value, μ_i, and is linearly decreased to some specified final value, μ_f, as the generations continue to allow the algorithm to make more specific changes after initially covering a very wide search space. μ_i is by default 20% of the maximum possible coupling, such that for a three digit genome, each mutation could initially change by up to 20% of 999: meaning $\mu_i = 200$. However, this should be changed to be higher or lower if a system requires more or less extreme changes, respectively. μ_f is set to unity by default, but should also be increased if μ should remain higher throughout the optimisation. New genomes are generated in this manner until an entirely new generation is created to replace the old one.

The overall process then repeats for a large number of generations, with each iteration resulting in an increased average fitness score until either a target fitness is reached or the program reaches some maximum elapsed iteration. Unless otherwise stated, optimizations were run for 200 generations, each containing 1024 genomes. These values were chosen as they allow sufficient time/diversity for most systems of these sizes. Note that, for optimum parallel performance, the number of genomes should

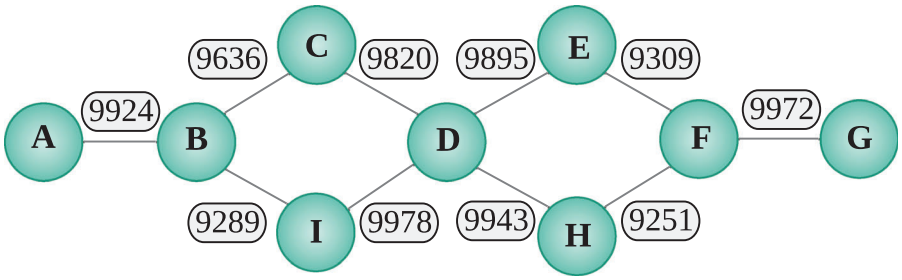

Figure 3. A network of spins arranged in a shoelace pattern optimized for speed and quantum state transfer. This maps $|1\rangle_A$ to $|1\rangle_G$ at time $t_f = 3.8/J_{\max}$ with 99.7% fidelity.

be chosen as a multiple of the number of CPU cores used to perform the optimization; thus powers of two work well.

4. Applications

We now provide examples of the application of the aforementioned algorithm for the design of different quantum devices. We will consider the on-site energies ϵ_i to be uniform and scaled to zero for the examples presented here.

4.1. Quantum State Distribution

As with classical computers and conventional data, a quantum computer processor requires quantum networks to be able to transmit quantum data between registers. Clearly the use of photonics for such short range communication presents some drawbacks: quantum computer hardware is generally built out of static matter qubits (e.g., superconductors, ion traps, or quantum dots) and the use of photons would imply the conversion of the matter qubit state into states of light and vice-versa, a costly process for such short distances. Instead, the idea of using linear spin chains as quantum data buses has attracted significant interest,[7,36–39] motivated by the possibility of building a "wire" with the same type of solid-state qubit as the rest of the hardware, to avoid conversion between different forms of qubits. One of the most well-known methods for doing this uses the natural dynamics of the system by engineering the spin–spin interactions of a one-dimensional chain.[7,38] Using only the natural dynamics implies that once the interactions are set, no further external control is required. We have used such existing results to both verify the accuracy of the algorithm (refer to Supporting Information[35]) and to test the algorithm's scaling (discussed in Section 5).

In this section, we focus on optimization of more complex, non-trivial networks for quantum state transfer and entanglement generation, for which our algorithm becomes more interesting. Here, a simple "shoelace" network was chosen as an example topology, initialized uniformly. When optimized for state transfer between sites A and G, it resulted in a transfer time of $t_f = 3.8/J_{\max}$ with 99.70% fidelity, with the resultant structure shown in **Figure 3**. This is 32% faster than a 7-site linear chain engineered with previously-known perfect state transfer capabilities,[38] and is attained by adding two extra nodes that modify the topology away from a linear chain. The identification of faster structures highlights the potential for this method to create improved spin-channels between quantum computing components.

Changing the parameters of the fitness function would result in a different result being converged upon, allowing flexibility depending on the physical implementation, for example, taking into account factors like the decoherence time. For instance, in the example above, an emphasis on shorter t_f was requested by changing the value of b (from Equation (3)) to -1000. Running with the standard value of $b = -0.001$ places more emphasis on the system's fidelity, resulting in a transfer time of $4.8/J_{\max}$, at a fidelity of 99.8%.

To further show the adaptability of this method, we perform optimizations on this same topology for various quantum information tasks, including examples of quantum state transfer tailored to occur at some chosen time, entanglement generation between arbitrary sites, and multi-excitation transfer. A summary of such tests is given in **Table 1**, all showing high fidelities. We note that, for tests in which specific times have not been specified, shorter transfer times may be requested at the cost of fidelity.

The best, average, and worst fitness scores for each generation during the optimization process leading to the values in Figure 3 are displayed in **Figure 4**, which shows how few generations are required for this method to reach a high fitness score. Note that here the optimization was stopped after the default maximum number of generations (200) as a demonstration, but could have been stopped much sooner and still retained fast and high fidelity state transfer. Corresponding graphs for the optimization of other systems are given in Supporting Information.[35] Importantly, although a simple method could be to take only the best genomes for each generation, the best fitness may then become trapped at a local maximum, while worse solutions may eventually reach an overall higher fitness if allowed to evolve down their path.

4.2. Design of a Quantum Gate

While transferring quantum information quickly and reliably is one of the most popular uses of spin networks, when it comes to designing quantum hardware there may be situations where it would significantly aid computation if a quantum gate could be applied to the information as it is being transferred. To do this with large spin networks, one would need to find suitable tuning of the numerous parameters, something which would be difficult to achieve analytically. We thus identify such an example as one of the most appropriate use cases for our proposed method.

Table 1. The various test optimizations performed on the shoelace topology (see Figure 3).

Task (Genome Notation)	Initial	Target	Fidelity	Time $\cdot J_{max}$
$\langle A\|G\rangle$ @5.0	$\|1\rangle_A$	$\|1\rangle_G$	99.7%	5.0
$\langle A\|G\rangle$ @8.0	$\|1\rangle_A$	$\|1\rangle_G$	99.8%	8.0
$\langle A\|E\rangle$	$\|1\rangle_A$	$\|1\rangle_E$	99.9%	3.8
$\langle C\|A+G\rangle$	$\|1\rangle_C$	$\frac{1}{\sqrt{2}}(\|10\rangle_{AG}+\|01\rangle_{AG})$	99.8%	3.0
$\langle C\|A-G\rangle$	$\|1\rangle_C$	$\frac{1}{\sqrt{2}}(\|10\rangle_{AG}-\|01\rangle_{AG})$	99.9%	3.4
$\langle C+H\|I+E\rangle$	$\frac{1}{\sqrt{2}}(\|10\rangle_{CH}+\|01\rangle_{CH})$	$\frac{1}{\sqrt{2}}(\|10\rangle_{IE}+\|01\rangle_{IE})$	99.9%	4.8
$\langle AB\|FG\rangle$	$\|11\rangle_{AB}$	$\|11\rangle_{FG}$	90.0%	4.4

This shows how our method is capable of finding solutions to many different tasks. Note each of these represents a different coupling scheme optimized for that purpose, not a single coupling scheme achieving all tasks. Each of these optimizations used standard parameters, with runtimes taking between 5 and 30 s using eight cores.[40]

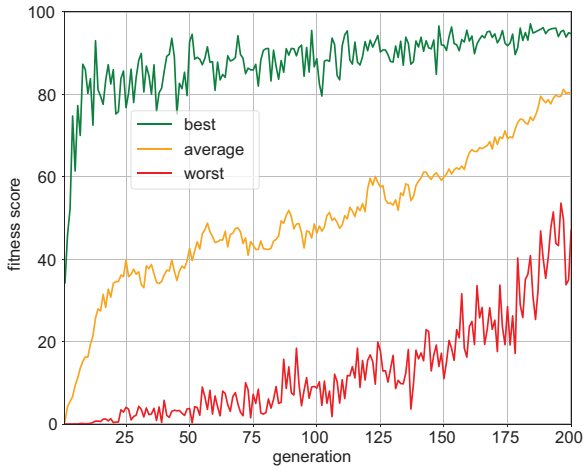

Figure 4. Worst, average, and best fitness scores $f(F_{max}, t_f)$ for each generation when optimizing a shoelace network for quantum state transfer, generating the result shown in Figure 3. The size of mutations is reduced each generation, resulting in smaller, more precise changes in fitness, except for the worst fitness scores, which are the product of more diverse genomes. Running all 200 generations has a runtime of around 5 s using eight cores.[40]

In the following test, we consider a 4×4 grid topology as a "blank canvas" for larger systems, plus two input and two output spins, with the aim of engineering the system to perform a controlled-Z gate on two qubits (see **Table 2**). In **Figure 5**, we draw the topology along with the optimized couplings: the gate is to be applied between qubits R and A, with the result being output at sites S and F. A key challenge in this application is the constraint that all input/output operations in the corresponding truth table (first three lines of Table 2) must be achieved in the same output time, and, of course, all with high fidelity.

The genetic algorithm was capable of identifying the tuning outlined in Figure 5 with the device able to perform a controlled-Z gate, allowing for the initial product state

$$|\Psi(0)\rangle = \frac{1}{2}(|00\rangle_{RA}+|01\rangle_{RA}+|10\rangle_{RA}+|11\rangle_{RA}) \tag{4}$$

to be mapped to the approximate final state of

$$|\Psi(t_f)\rangle \approx \frac{1}{2}(|00\rangle_{SF}+|01\rangle_{SF}+|10\rangle_{SF}-|11\rangle_{SF}) \tag{5}$$

with 99.8% fidelity and a transfer time of $t_f = 12.40/J_{max}$.

Table 2. Truth table showing the fidelities when the controlled-phase-gate genome is evaluated with different input injections and with various levels of genome precision, given as the number of significant figures (s.f.) used per coupling.

Initial	Final	4 s.f.	3 s.f.	2 s.f.	1 s.f.
$\|01\rangle_{RA}$	$\|01\rangle_{SF}$	99.8%	99.9%	99.8%	89.0%
$\|10\rangle_{RA}$	$\|10\rangle_{SF}$	99.9%	99.9%	99.9%	91.9%
$\|11\rangle_{RA}$	$-\|11\rangle_{SF}$	99.8%	99.8%	99.6%	83.8%
$\frac{1}{2}(\|00\rangle_{RA}+\|01\rangle_{RA}+\|10\rangle_{RA}+\|11\rangle_{RA})$	$\frac{1}{2}(\|00\rangle_{SF}+\|01\rangle_{SF}+\|10\rangle_{SF}-\|11\rangle_{SF})$	99.8%	99.8%	99.8%	89.0%

Here, by "n significant figures" we mean rounding each coupling to the nearest 10^{4-n}, for example, 1432 → 1000 for $n = 1$. Note that all sites are assumed to have no excitation unless otherwise specified (such that $|0\rangle$ states of non-relevant sites are omitted for clarity). The system is tailored so that each of these outputs is achieved at the same time of $12.4/J_{max}$.

Importantly, this network is shown to retain high ≈ 99.8% fidelity even as the number of digits used in the genome is approximated from 4 to 3 and even 2 significant figures (s.f.), which would allow tolerance when implementing such a network in the lab, as shown in Table 2. The approximation is done such that, for example, the 2 s.f. couplings are the 4 s.f. couplings, but rounded to the nearest 100 and then divided by 100 (e.g. 23 → 0, 2524 → 25 etc.). The fidelity for the genome with these 2 s.f. couplings is then evaluated and reported in Table 2. Even when the couplings are rounded up just to the nearest 1000 (right-end column in Table 2), the resulting approximated solution still retains a very high fidelity for the requested task. This implies that the minimum requirement for experiments is quite modest, that is, to be able to vary coupling energies between a set reference value (J_{max}) and a tenth of it, a modest requirement. While we cannot claim it to be a general result, we found that this robustness is shared by various other examples. This high tolerance would also suggest that to improve the performance of this particular network a change in the topology is needed, rather than simply increasing precision of the genome.

In this example, unlike the others, the two-excitation coupling term in Equation (1) affects the results when $\alpha \neq 0$. This term helps to build a phase specific to subspaces containing at least two excitations, allowing the network for the controlled phase gate to be optimized to reach higher fidelities. Without such a term, this

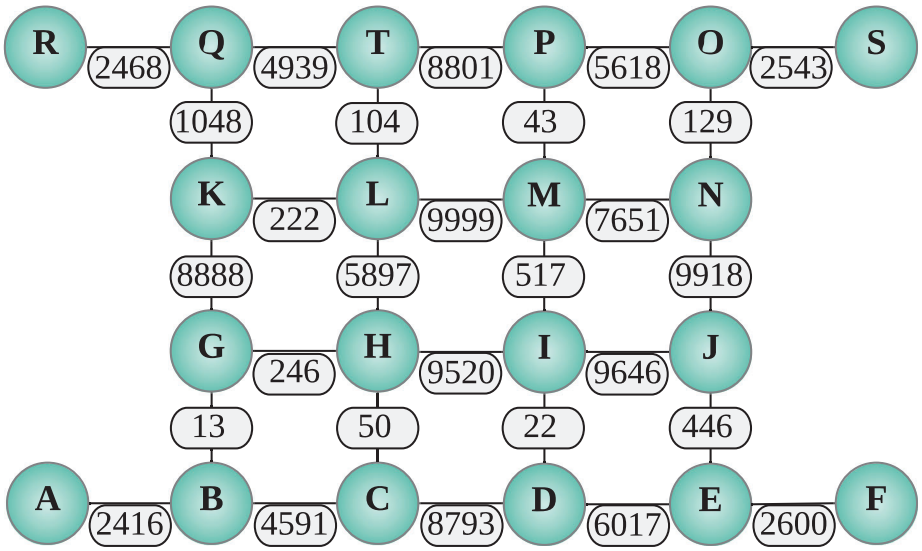

Figure 5. A network optimized to perform a controlled phase gate on the two qubits, and mapping the state $\frac{1}{2}(|00\rangle_{RA} + |10\rangle_{RA} + |01\rangle_{RA} + |11\rangle_{RA})$ to $\frac{1}{2}(|00\rangle_{SF} + |10\rangle_{SF} + |01\rangle_{SF} - |11\rangle_{SF})$ with 99.8% fidelity at time $t_f = 12.40/J_{\max}$.

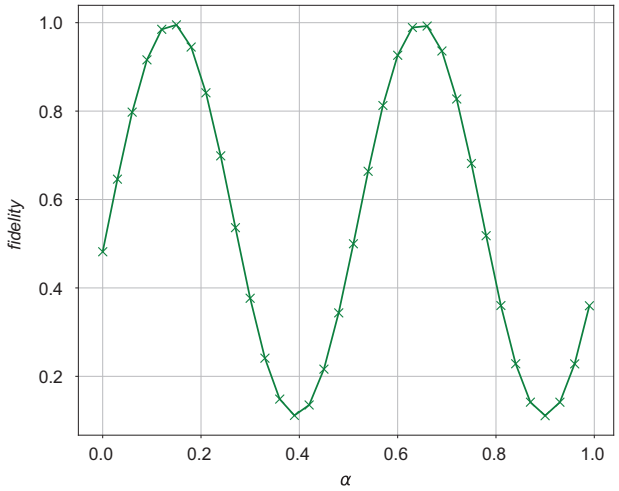

Figure 6. Fidelity $F(t_f)$ of the phase-gate versus α. The fidelity shows sinusoidal behavior with respect to α. The first peak is reached at $\alpha = 0.141$ with a fidelity of 99.8%.

topology was able to reach at most 77% fidelity, while optimizing with $\alpha = 0.141$ allowed for the 99.8% fidelity result. In some physical implementations, this coupling term would correspond to a dipole–dipole interaction. A scaling factor of $\alpha = 0.141$ with respect to each coupling J_{ij} is then consistent with this second-order type of interaction.

Figure 6 shows how the fidelity is affected when the phase-gate coupling scheme is evaluated using different values of α. It shows that the gate design is robust against small variation of α about its best value. The fidelity describes a sinusoidal variation with respect to α, confirming that this term is responsible for the creation of a phase, and hence offering multiple choices of α for achieving best fidelity.

5. Parallelization and Scaling

An interesting property of genetic algorithms is that they can be highly parallelized. In our case, we have been able to efficiently parallelize our algorithm using a standard implementation of the Message Passing Interface (MPI). This allowed large networks with multiple excitations to be fully optimized within an hour, which otherwise would have taken a day. This is all done by distributing the evaluation of each generation between the CPU cores, providing embarrassingly-parallel speedup.

The parallel performance of the code is shown in **Figure 7**, which shows how the time taken to optimize a system is reduced by a factor of approximately two using two cores, four with four cores, etc., a concept known as parallel speedup, ideally an identity function. The optimizations used for this test were run for a fixed number of generations (here 200) to focus more on the efficiency of the algorithm than on the ease of optimizing each given system. A positive feature is that the scaling is better for larger systems, since more time is spent evaluating each genome, a task done entirely in parallel, compared to smaller systems in which most of the time is spent on the more trivial serial operations, such as distributing/collecting the genomes between cores.

We also consider how the algorithm scales as the systems become larger. This was done by extending a linear chain and timing how long was required to reach 90%, 95%, and 97% fidelity for an end-to-end state transfer. The results are given in **Figure 8** and show that even for large systems of 32 qubits, and using just four cores, the optimizations are still performed in under a minute to a very high fidelity and in only a few seconds to good fidelity[41]. All optimizations here use the same algorithm parameters (such as the number of genomes or the maximum mutation size) to allow for a fairer comparison. If improved performance is desired, these parameters should be manually tailored for each system.

For the system shown in Figure 8, the time t_{opt} required to optimize such a system is shown to scale exponentially with

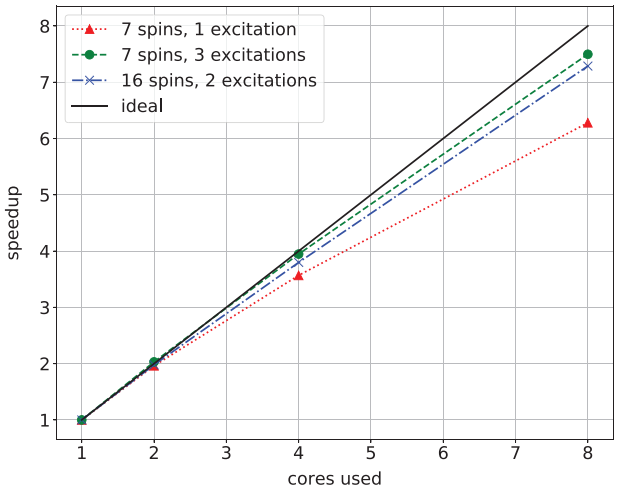

Figure 7. Scaling of the algorithm for a given system as the number of CPU cores used in parallel is increased. Here, speedup $= t_1/t_n$, with t_n the time for a certain optimization using n cores. Three systems are used for demonstration: a linear chain of 7 sites using a single excitation, the same chain but with three excitations, and a 16 spin grid with two excitations. Each is optimized using 1024 genomes for 200 generations. Each point was averaged over five realisations[40]. Note how scaling becomes closer to the ideal case as the subspaces become larger, allowing more efficient CPU usage.

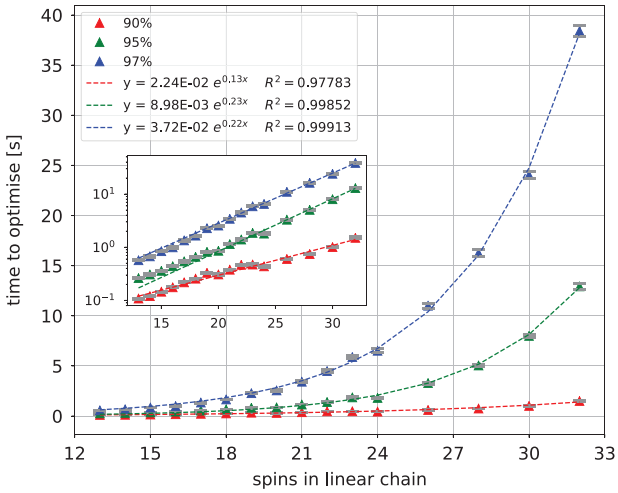

Figure 8. Scaling of the algorithm as the system size increases, specifically for a linear chain optimized to transfer fidelities of 90%, 95%, and 97%. Each point was averaged over 400 realizations, each done in parallel using four cores[41]. These optimizations all use the same parameters for a fair comparison; however, performance can be improved by tweaking parameters on a case-by-case basis.

the system size, albeit with very small coefficients, for example, $t_{opt} = 4 \times 10^{-2} \exp(0.22N)$ for 97% fidelity and N spins. This is as expected, since the search space grows exponentially with every added coupling (one coupling has 9999 possibilities, two have a total of 9999^2, and so on). Further analysis on the algorithm's scaling is given in Supporting Information, where the algorithm is run for a fixed number of generations (200), but for different numbers of excitations, to more directly show the effect of increasing the size of the Hilbert space[35].

While we cannot compare the efficiency of our method with previous approaches due to a lack of benchmarks, we can study how it compares to a randomized search of the parameter space. Our results show that even for a modest 10 qubit linear chain (9 parameters to be optimized), the randomized approach reached 99% fidelity only after 48 min of parallelized searching, while our genetic algorithm converges 5000 times faster for that same example. If we now move to a larger system, such as our proposed phase gate (28 parameters to be optimized), it would be completely infeasible to find appropriate solutions either randomly or analytically due to the sheer size of the parameter space. Also, our 5000 times speedup against the randomized approach is in contrast with the one obtained by another piece of research,[30] where their machine learning algorithm yields optimization times 180 times faster than the automated random search of the parameter space, although a direct comparison is difficult due to the different nature of the two problems.

6. Conclusions

As quantum devices improve in terms of number of qubits and connectivity, the power of quantum computation gets an exponential boost. However, this comes with the challenge that an increased number of degrees of freedom brings to the tuning of parameters when engineering quantum devices. To overcome this, we propose a novel method based on evolutionary computation that is capable of efficiently finding appropriate solutions from within the large search space of possible parametrizations. This method exploits a genetic algorithm that we have specifically designed to optimize the different degrees of freedom of a spin network to perform any given quantum information task. We provide examples showing new network designs discovered by the algorithm, acting as a controlled phase gate on two qubits, an entangler, or allowing for fast and high fidelity quantum information transfer between arbitrary sites of a network. Such networks are shown to retain high fidelities even when the precision on the coupling energies is reduced, which would allow a margin for fabrication and control errors when implementing such devices experimentally. To demonstrate flexibility, we have used some example networks with non-trivial topologies, where multiple paths exist between points of interest in the network.

It is important to note that our method allows optimization with a tailored fitness function as well as the inclusion of a set of optional and flexible parameters. This versatility provides the possibility of a variety of use cases depending on the specific experimental constraints. For example, when designing a quantum state transfer device, it may be preferable to transfer information faster at the expense of a lower fidelity in cases where decoherence times are relatively short. Our algorithm can be programmed with such constraints, making our method a promising candidate to assist in the design and calibration of real NISQ devices. Due to the highly parallelizable nature and efficiency of genetic methods, the algorithm converges rapidly, often within only seconds, with convergence occurring approximately 5000 times faster than the equivalent randomized search for a 10-parameter network. It also scales well with an increasing number of spins and excitation subspaces, with trends that, even though exponential, present small coefficients: very good solutions are found after just a small number of generations and thus

in a relatively short time; for example, less than 40 *s* are needed for optimizing a 32-parameters spin chain to 97% fidelity.

The method we propose can be easily extended to include additional terms in the Hamiltonian, or to different model Hamiltonians. Future research will involve testing the method against a larger set of initial topologies and quantum information tasks, such as optimising for different quantum gates. Although our code allows for the optimization of the on-site energies, in this work, it was decided to focus on optimizing only the coupling values, to allow us to compare with more well known results. However, future investigations that also include on-site energies are clearly of interest, not only both from a theory and modeling perspective, but also with respect to physical implementations. Further investigations will also involve extending our methods to provide results bound by the experimental constraints, for example, given by a specific quantum chip implementation.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

M.P.E. would like to acknowledge support from the Japanese MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant Number JPMXS0118069605. This project was in part undertaken on the Viking Cluster, a high-performance computing facility provided by the University of York. The authors are grateful for the computational support from the University of York High Performance Computing service, Viking and the Research Computing team.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are openly available in The York Research Database at https://doi.org/10.15124/5f1839f2-a4a7-4318-8a38-fade318507f8.

Keywords

evolutionary computation, genetic algorithms, spin chains, spin networks, quantum computing, quantum information processing, quantum devices

Received: January 28, 2021
Revised: April 27, 2021
Published online: June 15, 2021

[1] J. Preskill, Quantum Computing and the Entanglement Frontier, *arXiv:1203.5813*, **2012**.
[2] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, et al., *Nature* **2019**, *574*, 7779.
[3] S.-K. Liao, W.-Q. Cai, W.-Y. Liu, L. Zhang, Y. Li, J.-G. Ren, J. Yin, Q. Shen, Y. Cao, Z.-P. Li, F.-Z. Li, X.-W. Chen, L.-H. Sun, J.-J. Jia, J.-C. Wu, X.-J. Jiang, J.-F. Wang, Y.-M. Huang, Q. Wang, Y.-L. Zhou, L. Deng, T. Xi, L. Ma, T. Hu, Q. Zhang, Y.-A. Chen, N.-L. Liu, X.-B. Wang, Z.-C. Zhu, C.-Y. Lu, et al., *Nature* **2017**, *549*, 7670.
[4] S. Wehner, D. Elkouss, R. Hanson, *Science* **2018**, *362*, 6412.
[5] G. Kurizki, P. Bertet, Y. Kubo, K. Mølmer, D. Petrosyan, P. Rabl, J. Schmiedmayer, *Proc. Natl. Acad. Sci. USA* **2015**, *112*, 13.
[6] G. M. Nikolopoulos, I. Jex (Eds.), *Quantum State Transfer and Network Engineering*, Springer, New York **2014**.
[7] M. Christandl, N. Datta, A. Ekert, A. J. Landahl, *Phys. Rev. Lett.* **2004**, *92*, 18.
[8] G. M. Nikolopoulos, D. Petrosyan, P. Lambropoulos, *J. Phys.: Condens. Matter* **2004**, *16*, 28.
[9] Y. Wang, F. Shuang, H. Rabitz, *Phys. Rev. A* **2011**, *84*, 1.
[10] P. Karbach, J. Stolze, *Phys. Rev. A* **2005**, *72*, 3.
[11] L. Vinet, A. Zhedanov, *Phys. Rev. A* **2012**, *85*, 012323.
[12] M. P. Estarellas, I. D'Amico, T. P. Spiller, *Sci. Rep.* **2017**, *7*, 42904.
[13] A. Blanco-Redondo, I. Andonegui, M. J. Collins, G. Harari, Y. Lumer, M. C. Rechtsman, B. J. Eggleton, M. Segev, *Phys. Rev. Lett.* **2016**, *116*, 163901.
[14] S. Longhi, *Phys. Rev. B* **2019**, *99*, 155150.
[15] I. D'Amico, B. W. Lovett, T. P. Spiller, *Phys. Rev. A* **2007**, *76*, 3.
[16] T. P. Spiller, I. D'Amico, B. W. Lovett, *New J. Phys.* **2007**, *9*, 20.
[17] M. P. Estarellas, I. D'Amico, T. P. Spiller, *Phys. Rev. A* **2017**, *95*, 4.
[18] T. J. G. Apollaro, G. M. A. Almeida, S. Lorenzo, A. Ferraro, S. Paganelli, *Phys. Rev. A* **2019**, *100*, 052308.
[19] R. Ronke, I. D'Amico, T. Spiller, *Phys. Rev. A* **2011**, *84*, 3.
[20] Y. Tserkovnyak, D. Loss, *Phys. Rev. A* **2011**, *84*, 3.
[21] A. Landahl, M. Christandl, N. Datta, A. Ekert, *AIP Conf. Proc.* **2004**, *734*, 215.
[22] A. E. Eiben, J. Smith, *Nature* **2015**, *521*, 7553.
[23] A. E. Eiben, J. E. Smith, *Introduction to Evolutionary Computing*, Springer, New York **2003**.
[24] G. Hornby, A. Globus, D. Linden, J. Lohn, "Automated Antenna Design with Evolutionary Algorithms," in: *SPACE 2006*, San Jose, CA, Sept. 2006, American Institute of Aeronautics and Astronautics (AIAA), Reston, VA **2012**, https://doi.org/10.2514/6.2006-7242.
[25] B. Evans, S. Walton, *Appl. Math. Modell.* **2017**, *52*, 215.
[26] X. Yao, *Proc. IEEE* **1999**, *87*, 9.
[27] M. Schmidt, H. Lipson, *Science* **2009**, *324*, 5923.
[28] D. Whitley, *Stat. Comput.* **1994**, *4*, 2.
[29] F. Domínguez-Serna, F. Rojas, *J. Phys. Conf. Ser.* **2015**, *624*, 012009.
[30] H. Moon, D. T. Lennon, J. Kirkpatrick, N. M. van Esbroeck, L. C. Camenzind, L. Yu, F. Vigneau, D. M. Zumbühl, G. A. D. Briggs, M. A. Osborne, D. Sejdinovic, E. A. Laird, N. Ares, *Nat. Commun.* **2020**, *11*, 4161.
[31] N. van Esbroeck, D. Lennon, H. Moon, V. Nguyen, F. Vigneau, L. Camenzind, L. Yu, D. Zumbühl, G. Briggs, D. Sejdinovic, N. Ares, *arXiv:2001.04409*, **2020**.
[32] R. Ronke, T. Spiller, I. D'Amico, *Phys. Rev. A* **2011**, *83*, 012325.
[33] S. M. Thede, *J. Comput. Sci. Coll.* **2004**, *20*, no. 1, 115.
[34] J. Struck, C. Ölschläger, R. Le Targat, P. Soltan-Panahi, A. Eckardt, M. Lewenstein, P. Windpassinger, K. Sengstock, *Science* **2011**, *333*, 6045.
[35] See Supporting Information for additional examples of algorithm scaling, examples of optimization processes, and visualization of the genome.
[36] S. Bose, *Phys. Rev. Lett.* **2003**, *91*, 20.
[37] S. Bose, *Contemp. Phys.* **2007**, *48*, 13.
[38] M. Christandl, N. Datta, T. C. Dorlas, A. Ekert, A. Kay, A. J. Landahl, *Phys. Rev. A* **2005**, *71*, 3.
[39] A. Kay, *Int. J. Quantum Inf.* **2010**, *8*, 641.
[40] Simulated with a desktop computer with an AMD Ryzen 7 3700X.
[41] Simulated on a laptop with an Intel(R) Core(TM) i7-6700HQ CPU.

